VERİDEN ANLAMA

Bilgi Grafları, Semantik Sistemler ve Yapay Zekâ Ekosistemleri

Necmi Gürsakal & ChatGPT

Bursa-2026

VERİDEN ANLAMA

Bilgi Grafları, Semantik Sistemler ve Yapay Zekâ Ekosistemleri

NECMİ GÜRSAKAL & CHATGPT

Bursa-2026

ARKA KAPAK YAZISI

Veri artık eksik değil. Aksine, çağımızın temel sorunu veri kıtlığı değil; veri fazlalığı içinde anlamın kaybolmasıdır. Kurumlar, sistemler ve bireyler her saniye devasa miktarda veri üretirken, bu verilerin birbirinden kopuk yapısı nedeniyle dünya giderek daha karmaşık ama daha az anlaşılır hale gelmektedir.

“Veriden Anlama: Bilgi Grafları”, bu karmaşık dünyayı yeniden okunabilir kılmak için geliştirilmiş bütüncül bir düşünme çerçevesi sunar. Kitap, veriyi yalnızca teknik bir nesne olarak değil, ilişkiler ağı içinde anlam kazanan bir yapı olarak ele alır. Bu nedenle merkezinde yalnızca veri tabanları değil; bilgi grafı düşüncesi, semantik web, ontolojiler ve modern yapay zekâ mimarileri yer alır.

Eserde, klasik veri tabanı anlayışından modern graf veri modellerine geçiş ayrıntılı biçimde incelenir. Neo4j gibi graf veri tabanlarının yanı sıra RDF, OWL ve SPARQL gibi semantik web teknolojileri de sistematik bir şekilde ele alınır. Ancak kitap yalnızca teknik bir referans değildir; aynı zamanda bir düşünme biçimi önerir: Dünyayı tablolarla değil, ağlarla anlamak.

Bu yaklaşım, yalnızca veri mühendisliği alanına değil; yapay zekâdan sosyal ağ analizine, finansal sistemlerden savunma sanayine kadar geniş bir uygulama alanına uzanır. Kitapta yer alan Palantir Gotham ve Foundry bölümleri, bilgi grafı yaklaşımının gerçek dünyadaki operasyonel sistemlerde nasıl kullanıldığını göstererek teoriyi somut uygulamalarla birleştirir.

“Veriden Anlama”, aynı zamanda yapay zekâ çağının temel bir sorusuna odaklanır: Bir makine veriyi yalnızca işleyebilir mi, yoksa onu anlamlandırabilir mi? Bu soru üzerinden bilgi grafı tabanlı sistemlerin, büyük dil modelleri (LLM), RAG mimarileri ve graf sinir ağları (GNN) ile nasıl birleştiği tartışılır.

Kitap, okuyucuyu yalnızca teknik araçlarla tanıştırmakla kalmaz; veri, bilgi ve anlam arasındaki sınırları yeniden düşünmeye davet eder. Çünkü modern dünyada asıl dönüşüm, daha fazla veri üretmekte değil, veriler arasındaki ilişkileri görebilmektedir.

Bu eser, veri bilimi, yapay zekâ, bilgi sistemleri ve dijital dönüşüm alanlarında çalışan araştırmacılar, mühendisler ve düşünürler için hem teorik hem de kavramsal bir referans niteliğindedir.

Sonuç olarak bu kitap, tek bir iddia üzerine kuruludur:

Dünyayı anlamanın yeni yolu, nesneleri değil ilişkileri okumaktır.

İÇİNDEKİLER

ÖNSÖZ

GİRİŞ

BÖLÜM 1 — Bilgi grafı Nedir?
1.1 Veri, Enformasyon, Bilgi ve Anlam
1.2 Graf Düşüncesinin Tarihi
1.3 Düğüm (Node), Bağlantı (Edge), İlişki (Relation)
1.4 Ontoloji Nedir?
1.5 Semantik Web'in Doğuşu
1.6 RDF, OWL ve SPARQL'a Genel Bakış
1.7 Bilgi Grafı mı, Graf Veri Tabanı mı?
1.8 Neo4j, TigerGraf, Amazon Neptune ve Diğerleri
1.9 Günlük Hayatta Bilgi grafı Örnekleri
BÖLÜM 2 — Graflarla Düşünmek
2.1 İnsan Zihni ve İlişkisel Düşünme
2.2 Ağ Bilimi ve Graf Kuramı
2.3 Küçük Dünya Ağları
2.4 Ölçekten Bağımsız Ağlar
2.5 Merkezilik Kavramı
2.6 Topluluklar ve Kümeleşme
2.7 Bilgi Akışı ve Ağ Dinamikleri
2.8 Sosyal Ağlardan Bilgi Ağlarına
2.9 Graf Düşünme: Dünyayı İlişkilerle Okumak
BÖLÜM 3 — Bilgi Grafı Mimarisi
3.1 Üçlü Yapılar: Özne–Yüklem–Nesne
3.2 Şema Tasarımı
3.3 Ontoloji Geliştirme Süreci
3.4 Varlık Eşleştirme Problemi
3.5 Veri Bütünleştirme
3.6 Üstveri ve Anlamsal Katman
3.7 Bilgi Grafı İşlem Hattı
3.8 Gerçek Zamanlı Bilgi Grafı Sistemleri
3.9 Graf Destekli Sorgu ile Üretim Mimarileri (Graf Retrieval-Augmented Generation Architectures / GrafRAG)
BÖLÜM 4 — Graf Veri Tabanları ve Neo4j
4.1 Neden Yeni Bir Veri Modeline Gerek Duyuldu?
4.2 Graf Düşüncesi: Veri Değil İlişki Merkezlidir
4.3 Graf Veri Modelleri
4.4 Neo4j: Graf Tabanlı Veri Tabanı
4.5 Cypher: Graf ile Konuşmak
4.6 Neo4j'ye Veri Girişi
4.7 Neo4j ile İlk Uygulama: Tedarik Zinciri Grafı
4.8 Graf Algoritmaları ve Analitik
4.9 Neo4j ve Bilgi Grafı Dünyası

4.10 Neo4j + Yapay Zekâ
4.11 Sonuç: Veritabanından Anlam Motoruna
BÖLÜM 5 — Bilgi Grafı ve Yapay Zekâ
5.1 Yapay Zekâ Neden Bilgiye İhtiyaç Duyar?
5.2 Büyük Dil Modelleri (LLM) ve Bilgi Grafı Birleştirme
5.3 Bilgi Getirmeli Üretim (Retrieval-Augmented Generation, RAG)
5.4 Graf Tabanlı RAG Yaklaşımı (GrafRAG)
5.5 Uydurma Bilgi Üretme (Hallucination) Problemi
5.6 Açıklanabilir Yapay Zekâ (ExplYZnable YZ) ve Anlamsal İzlenebilirlik
5.7 Bilgi Grafı Destekli Özerk Sistemler (Ajan Sistemleri)
5.8 Çoklu Ajan (Multi-Agent) Yapıları
5.9 Sinirsel-Sembolik Yapay Zekâ (Neuro-Symbolic YZ)
5.10 Graf Sinir Ağları (Graf Neural Networks, GNN)
5.11 Vektör Veri Tabanları ve Graf Veri Tabanları İlişkisi
5.12 Yapay Zekânın Belleği Olarak Bilgi Grafı
5.13 Üretken Algoritmik Sistemler ve AlphaEvolve
BÖLÜM 6 — Bilgi Grafı Oluşturma Teknikleri
6.1 Yapılandırılmamış Veriden Bilgi Çıkarma
6.2 Doğal Dil İşleme ile Varlık Çıkarımı
6.3 İlişki Çıkarımı
6.4 Gömme (Embedding) Tabanlı Bilgi Çıkarımı
6.5 Bilgi Füzyonu
6.6 Veri Temizleme ve Gürültü Problemi
6.7 Otomatik Ontoloji Üretimi
6.8 LLM Kullanarak Bilgi Grafı İnşası
6.9 PDF, Web ve Sosyal Medyadan Bilgi Toplama
6.10 Bilgi Güncelleme ve Evrim Problemi
BÖLÜM 7 — Uygulama Alanları
7.1 Google Bilgi Grafı
7.2 Arama Motorları ve Semantik Arama
7.3 Öneri Sistemleri
7.4 Finans ve Dolandırıcılık Analizi
7.5 Sağlık ve İlaç Keşfi
7.6 Endüstri 4.0 ve Dijital İkizler
7.7 Savunma ve İstihbarat Sistemleri
7.8 Akademik Bilgi Ağları
7.9 Akıllı Şehirler
7.10 Sosyal Ağ Analitiği
BÖLÜM 8 — Bilgi Grafı ve Bilim
8.1 Bilimsel Keşiflerde Ağ Yaklaşımı
8.2 Atıf Ağları ve Bilim Haritaları
8.3 Nobel Ödülleri ve Bilgi Ağları
8.4 Protein Etkileşim Ağları
8.5 Genetik ve Biyoinformatik
8.6 Fikirlerin Evrimi
8.7 Bilim İnsanları Arasındaki Gizli Bağlantılar
BÖLÜM 9 — Kodlama ve Projeler

9.1 Python ile Graf Veri Yapıları
9.2 NetworkX Kullanımı
9.3 Neo4j Kurulumu
9.4 İlk Cypher Sorguları
9.5 Kùzu Kurulumu ve Python API
9.6 PDF ve CSV Verilerinden Bilgi Grafı Oluşturma
9.7 Wikipedia'dan Bilgi Grafı Çıkarmak
9.8 ChatGPT + Bilgi Grafı Uygulaması
9.9 Akademik Makalelerden Bilgi Ağı Kurmak
9.10 Kurumsal Bilgi Grafı Tasarımı
9.11 Mini Proje: LLM Destekli Akademik Arama Motoru
9.12 Tek Kişilik Bilgi Grafı Laboratuvarı
9.13 Proje Atölyeleri
9.14 Bölüm Sonu: Koddan Anlama
BÖLÜM 10 — Gelecek
10.1 Web 3.0 ve Semantik İnternet
10.2 Dijital Hafıza Sistemleri
10.3 İnsanlığın Ortak Bilgi Grafı
10.4 Yapay Genel Zekâ ve Bilgi Temsili
10.5 Bilginin Politik Ekonomisi
10.6 Veri Sömürgeciliği ve Güç
10.7 Bilgi Grafı Çağında Üniversiteler
10.8 "Her Şeyin Grafı" Mümkün mü?
10.9 Simülasyon Uygarlığı ve Dünya Modeli
10.10 Kendini Yinelemeli Geliştirme (RSI)
BÖLÜM 11 — Palantir ve Operasyonel Bilgi Grafı Çağı
11.1 Silikon Vadisi'nden İstihbarat Dünyasına
11.2 Gotham: Dünyayı Bir Ağ Gibi Görmek
11.3 Foundry: Şirketlerin Dijital Sinir Sistemi
11.4 Ontoloji Kavramının Endüstride Kullanımı
11.5 Bilgi Grafı + YZ + Operasyon
11.6 Savunma Sanayi ve Bilgi Ağları
11.7 Sağlık, Enerji ve Üretimde Palantir Yaklaşımı
11.8 Palantir ve Yapay Zekâ Platformları
11.9 Project Maven ve Askeri YZ Ekosistemi
11.10 Büyük Güç Mü, Büyük Gözetim Mi?

EKLER

Ek A — Temel Graf Kuramı
Ek B — RDF ve OWL Hızlı Başvuru
Ek C — Cypher Cheat Sheet
Ek D — Açık Veri Kaynakları
Ek E — Türkçe Kaynaklar ve Topluluklar
Ek F — Modern Açık Kaynak Graf Ekosistemi
Ek G —Uçtan Uca Veri → Graf Dönüşümü Örnek Olayı

SÖZLÜK

Rahmetli Şevket Ertunga'nın anısına

ÖNSÖZ

İnsanlık tarihi biraz da bilgiyi düzenleme tarihidir. Kil tabletlere yazılan ticaret kayıtlarından kütüphanelere, ansiklopedilerden veri tabanlarına kadar her çağ kendi bilgi mimarisini üretmiştir. Bugün ise yeni bir dönüşümün içindeyiz: veriyi yalnızca depolayan değil, veriler arasındaki ilişkileri anlayan sistemlerin çağındayız. Bu dönüşümün merkezinde ise *Bilgi grafı* yani bilgi grafı yaklaşımı yer almaktadır.

Uzun yıllar boyunca bilgisayar sistemleri dünyayı tablolar halinde görmeye alıştı. Satırlar, sütunlar, anahtarlar ve kayıtlar… İlişkisel veri tabanları modern dijital ekonominin omurgasını oluşturdu. Bankalar, üniversiteler, hastaneler, devlet kurumları ve şirketler verilerini bu yapı üzerinde yönetti. Ancak dünya aslında tablo biçiminde değildir. İnsanlar, kurumlar, olaylar, kavramlar ve makineler; birbirleriyle sürekli ilişki içinde olan devasa ağlar oluştururlar. Gerçek dünya bir tablo değil, bir bağlantılar evrenidir.

İşte bilgi grafı yaklaşımı bu nedenle önemlidir. Çünkü bilgi grafı, yalnızca "veri saklama" yöntemi değildir; aynı zamanda dünyayı düşünme biçimidir. Bu yaklaşımda temel unsur tek tek veriler değil, veriler arasındaki anlamlı ilişkilerdir. Bir insanın başka bir insanla, bir şirketin bir teknolojiyle, bir bilim insanının bir keşifle veya bir şehrin enerji altyapısıyla kurduğu bağlar sistemin merkezine yerleşir. Böylece bilgisayarlar sadece kayıtları değil, bağlamı da anlamaya başlar.

Bugün arama motorlarından öneri sistemlerine, sosyal ağlardan ilaç keşfine kadar pek çok alanda bilgi grafları kullanılmaktadır. Google'ın "şeyleri anlamaya" çalışması, Amazon'un ürün ilişkilerini modellemesi, modern yapay zekâ sistemlerinin bilgi eksikliğini gidermek istemesi ve savunma sanayisinde gerçek zamanlı veri ilişkilendirme ihtiyacı; hep aynı noktaya işaret etmektedir: Bilgi artık yalnızca depolanmıyor, haritalanıyor.

Yapay zekânın yükselişi bu dönüşümü daha da hızlandırmıştır. Büyük dil modelleri inanılmaz derecede güçlü olsalar da çoğu zaman tutarlı bir bilgi yapısından yoksundur. Bu nedenle son yıllarda "LLM + Bilgi grafı" birleşimi yeni bir paradigma haline gelmiştir. Bilgi grafı sistemleri yapay zekâya hafıza, izlenebilirlik, bağlam ve açıklanabilirlik kazandırmaktadır. Başka bir ifadeyle, yapay zekânın sezgisel dili ile bilgi graflarının yapısal mantığı birleşmeye başlamıştır.

Bu kitap, bilgi grafı dünyasına hem kavramsal hem uygulamalı bir giriş sunmayı amaçlamaktadır. Ağ bilimi, semantik web, ontoloji geliştirme, graf veri tabanları, yapay zekâ entegrasyonu, GrafRAG sistemleri[1] ve gerçek dünya uygulamaları birlikte ele alınacaktır. Aynı

[1] GrafRAG (Graph Retrieval-Augmented Generation), büyük dil modellerinin (LLM) harici bir bilgi kaynağından yararlanarak daha doğru cevaplar üretmesini sağlayan RAG yönteminin, Bilgi Grafları (Knowledge Graphs) ile birleştirilmiş en gelişmiş versiyonudur. Geleneksel RAG sistemleri metinleri düz parçalara (vektörlere) bölüp ararken, GrafRAG veriler arasındaki ilişkileri bir ağ (network) şeklinde haritalandırarak arama yapar.

zamanda okuyucu; Python, Neo4j ve Kùzu gibi araçlarla kendi bilgi grafı sistemlerini kurabilecek uygulamalı örneklerle karşılaşacaktır.

Çünkü önümüzdeki yıllarda en önemli rekabet yalnızca “daha fazla veri toplamak” değil, veriden anlam üretebilmek olacaktır. Bu kitap, tam da bu yeni çağın eşiğinde, veriden anlama giden yolculuğa küçük bir katkı sunma amacı taşımaktadır.

GİRİŞ

İnternetin ilk yıllarında arama motorları büyük ölçüde kelimelerle çalışıyordu. Bir kullanıcı arama kutusuna birkaç sözcük yazıyor, sistem de bu sözcüklerin geçtiği sayfaları bulmaya çalışıyordu. Başlangıçta bu yaklaşım oldukça başarılıydı. Çünkü web henüz bugünkü kadar büyük, karmaşık ve çok katmanlı değildi. Ancak internet büyüdükçe yeni bir problem ortaya çıktı: İnsanlar kelimeler aramıyordu; insanlar anlam arıyordu.

Örneğin bir kullanıcı "Einstein" yazdığında aslında ne istiyordu? Albert Einstein'ın biyografisi mi? Görelilik kuramı mı? Nobel ödülü mü? Fotoğrafları mı? Yazdığı makaleler mi? Yoksa sadece doğum tarihi mi? Kelimeler tek başına yeterli değildi. Arama motorlarının artık "şeyleri" yani varlıkları, kavramları, insanları, olayları ve bunların ilişkilerini anlaması gerekiyordu.

Bu dönüşümün en önemli kilometre taşlarından biri, 2012 yılında Google tarafından duyurulan "Bilgi Grafı" yaklaşımı oldu. Google artık web sayfalarını yalnızca belge olarak değil, birbiriyle ilişkili varlıkların oluşturduğu devasa bir bilgi ağı olarak görmeye başlıyordu. Bir film oyuncusuyla, oynadığı filmler; bir bilim insanıyla, keşifleri; bir şehirle, tarihi olaylar arasında bağlar kuruluyordu.

Bu yaklaşım aslında internetin doğasına daha uygundu. Çünkü gerçek dünya tablo halinde değil, ağ biçiminde örgütlenmiştir. İnsan hayatı ilişkilerle ilerler. Dostluklar, şirket ortaklıkları, bilimsel işbirlikleri, ekonomik bağlantılar, kültürel etkiler ve teknolojik bağımlılıklar… Bunların hepsi bir ağ oluşturur. Bilgi grafı yaklaşımı tam olarak bu ilişkisel gerçekliği dijital ortama taşımaya çalışmaktadır.

Bugün modern yapay zekâ sistemlerinden öneri motorlarına, biyoinformatikten savunma sanayisine kadar çok geniş bir alanda aynı düşünce biçimi kullanılmaktadır: Dünyayı bir graf (çizge) olarak görmek.

Veriden Anlama Yolculuğu

Bilgisayar bilimlerinin ilk dönemlerinde en önemli mesele veriyi depolamaktı. Daha sonra veri yönetimi, veri erişimi ve veri güvenliği ön plana çıktı. İlişkisel veri tabanları bu dönemin en büyük devrimlerinden biriydi. Ancak zamanla veri miktarı o kadar büyüdü ki artık yalnızca depolamak yeterli olmamaya başladı.

Çünkü veri tek başına anlam üretmiyordu.

Bir şirkette milyonlarca müşteri kaydı bulunabilir. Bir hastanede devasa sağlık arşivleri olabilir. Bir üniversitede yüz binlerce akademik makale saklanabilir. Fakat asıl soru şudur: Bu veriler arasındaki gizli ilişkiler nasıl keşfedilecektir?

Bilgi grafı yaklaşımı burada devreye girer. Çünkü bilgi grafı sistemleri yalnızca veriyi değil, veriler arasındaki ilişkileri de modellemeye çalışır. Böylece sistem, yalnızca "kim?" veya "ne?" sorularını değil; "kiminle bağlantılı?", "hangi olayla ilişkili?", "hangi ağın parçası?" gibi daha derin soruları da cevaplayabilir.

Bu nedenle bilgi grafı teknolojileri yalnızca teknik araçlar değildir. Aynı zamanda yeni bir düşünme biçimidir. Bu yaklaşım, dünyayı bağımsız nesneler yerine ilişkiler ağı olarak görür. Aslında bu fikir yalnızca bilgisayar bilimlerine ait değildir. Sosyoloji, biyoloji, ekonomi ve nörobilim gibi pek çok alan uzun zamandır ağ yaklaşımını kullanmaktadır. Hücreler protein ağları oluşturur. İnsanlar sosyal ağlar kurar. Bilim insanları atıf ağları içinde çalışır. Şirketler tedarik zinciri ağlarına bağlıdır. Modern dünya giderek daha fazla "ilişkisel" hale gelmektedir.

Son yıllarda yapay zekâ araştırmalarında giderek daha önemli hale gelen kavramlardan biri de "Kendini Yinelemeli Geliştirme" (Recursive Self-Improvement, RSI) fikridir. Bu yaklaşımın temel sorusu şudur: Bir yapay zekâ sistemi yalnızca dünyayı modellemekle mi kalacaktır, yoksa kendi öğrenme süreçlerini de geliştirebilecek midir?

Bugünkü büyük dil modelleri ve modern yapay zekâ sistemleri olağanüstü ölçekte veri işleyebilmekte, metin üretebilmekte ve karmaşık problemleri çözebilmektedir. Ancak bu sistemlerin büyük çoğunluğu hâlâ insanlar tarafından tasarlanmış mimariler içinde çalışmaktadır. Öğrenme hedefleri, eğitim süreçleri ve temel algoritmalar dışarıdan belirlenmektedir.

Buna rağmen yapay zekâ alanındaki gelişmeler yeni bir ihtimali gündeme getirmektedir: Eğer bir sistem yalnızca bilgi üretmekle kalmayıp, kendi öğrenme yöntemlerini de geliştirmeye başlarsa ne olacaktır? İşte RSI kavramı tam olarak bu düşünsel sınırda ortaya çıkar.

Bu fikir bugün hâlâ büyük ölçüde teorik ve tartışmalıdır. Ancak modern yapay zekâ araştırmaları giderek daha üretken sistemlere doğru ilerlemektedir. Artık bazı sistemler yalnızca veri analiz etmekle kalmamakta; yeni algoritmalar önermekte, kod yazmakta, araştırma süreçlerine katkı sağlamakta ve karmaşık karar döngülerini yönetebilmektedir.

Bu gelişme, bilgi grafı yaklaşımı açısından da önemlidir. Çünkü bilgi grafı sistemleri yalnızca veriyi depolayan yapılar değildir; aynı zamanda makinelerin dünya hakkında daha kalıcı, ilişkisel ve anlamlı temsiller kurmasına yardımcı olur. Eğer geleceğin yapay zekâ sistemleri kendi düşünme süreçlerini geliştirmeye yaklaşacaksa, bunu büyük ölçüde anlam katmanı güçlü bilgi mimarileriyle birlikte yapmaları beklenebilir. Bu nedenle bilgi grafı teknolojileri yalnızca veri yönetimi açısından değil, geleceğin yapay zekâ mimarileri açısından da stratejik bir önem taşımaktadır.

İlginç bir şekilde RSI benzeri süreçler yalnızca yapay sistemlere özgü değildir. İnsan biyolojisi de belirli düzeylerde kendini yinelemeli geliştirme mekanizmaları içerir. Örneğin insan öğrenmesi, sinir plastisitesi üzerinden sürekli olarak kendi yapısını yeniden düzenler. Bir beceri tekrarlandıkça ilgili sinir bağlantıları güçlenir, kullanılmayan bağlantılar zayıflar ve beynin bilgi işleme biçimi fiziksel olarak değişir. Bu süreç, bir anlamda öğrenme mekanizmasının kendi kendini yeniden şekillendirmesi olarak görülebilir.

Daha geniş bir perspektiften bakıldığında bağışıklık sistemi de benzer bir RSI benzeri dinamik sergiler. Vücut, karşılaştığı yeni patojenlere karşı yalnızca tepki vermekle kalmaz; aynı zamanda bu karşılaşmalardan öğrenerek gelecekteki savunma mekanizmalarını da kalıcı olarak günceller. Bu durum, biyolojik sistemlerde "deneyim üzerinden kendini güncelleme" ilkesinin varlığına işaret eder.

Ancak burada kritik fark şudur: İnsan biyolojisindeki bu kendini güncelleme süreçleri bilinçli bir tasarımın ürünü değildir ve doğrudan sistemin tüm öğrenme algoritmasını yeniden yazmaz. Buna karşılık yapay RSI, teorik olarak öğrenme mekanizmasının kendisini de yeniden tasarlayabilme iddiasını taşır. Bu ayrım, biyolojik adaptasyon ile yapay özyineli gelişim arasındaki sınırı belirleyen temel noktadır.

İnsan Beyni ile Bilgi Grafı Arasındaki Benzerlikler

İnsan zihni de büyük ölçüde ilişkisel çalışır. Beynimiz bilgileri bağımsız kutular halinde depolamaz. Kavramlar arasında bağlantılar kurarız. "Paris" kelimesini duyduğumuzda yalnızca bir şehir düşünmeyiz. Fransa, Eyfel Kulesi, sanat, moda, Seine Nehri, Fransız Devrimi ve hatta kişisel anılarımız aynı anda zihnimizde canlanabilir. Çünkü insan belleği ağ yapısına benzer biçimde çalışır.

Nörobilim araştırmaları, beynin milyarlarca nöron arasındaki bağlantılar üzerinden işlediğini göstermektedir. Bir düşünce başka düşünceleri tetikler. Bir kavram başka kavramlarla ilişki içinde anlam kazanır. İnsan zekâsının önemli bir bölümü tam da bu ilişki kurma kapasitesinden doğar.

Bilgi grafı sistemleri de benzer bir mantık taşır. Bir düğüm başka düğümlere bağlanır. İnsanlar şirketlerle, şirketler teknolojilerle, teknolojiler araştırmalarla ilişkilendirilir. Böylece tek tek veri parçaları daha büyük bir anlam yapısının parçası haline gelir.

Bu nedenle bazı araştırmacılar bilgi grafı yaklaşımını "makineler için yapısal hafıza" olarak görmektedir. Özellikle büyük dil modellerinin yükselişiyle birlikte bu fikir daha önemli hale gelmiştir. Çünkü modern yapay zekâ sistemleri çok güçlü olsalar da çoğu zaman kalıcı, izlenebilir ve mantıksal bir bilgi yapısından yoksundur. Bilgi grafları bu eksikliği gidermeye aday yapılardan biridir.

SQL Dünyasından Graf Dünyasına

Uzun yıllar boyunca yazılım dünyasının merkezinde SQL[2] ve ilişkisel veri tabanları vardı. Gerçekten de bu sistemler olağanüstü başarılı oldu. Finans sistemleri, muhasebe altyapıları, rezervasyon sistemleri ve kurumsal uygulamalar büyük ölçüde bu mimari üzerinde gelişti.

Ancak ilişkisel veri tabanlarının temel varsayımı şuydu: Veri büyük ölçüde tablolarda tutulabilir ve ilişkiler sınırlı sayıda join işlemleriyle[3] yönetilebilir.

Modern dünyanın karmaşıklığı arttıkça bu yaklaşım bazı alanlarda yetersiz kalmaya başladı. Özellikle sosyal ağlar, dolandırıcılık analizleri, öneri sistemleri ve bilgi keşfi gibi alanlarda ilişki sayısı çok hızlı büyüyordu. Bir düğümün binlerce hatta milyonlarca başka düğümle bağlantılı olduğu sistemlerde klasik tablo mantığı hantallaşmaya başladı.

Graf veri tabanları tam bu noktada yeni bir paradigma sundu. Burada merkezde tablolar değil ilişkiler vardı. Verinin kendisi kadar bağlantılar da birinci sınıf vatandaş haline geldi.

Bugün Neo4j, Kùzu, TigerGraf ve benzeri sistemler bu yeni yaklaşımın farklı örneklerini sunmaktadır. Bazıları büyük kurumsal operasyonlar için geliştirilirken, bazıları bireysel araştırmacılar ve yerel yapay zekâ sistemleri için optimize edilmektedir.

Önümüzdeki yıllarda SQL dünyası tamamen ortadan kalkmayacaktır. Ancak graf düşüncesinin giderek daha önemli hale geleceği açıktır. Çünkü modern dünyanın temel problemi artık yalnızca veri depolamak değil, ilişkileri anlamaktır.

Bu Kitap Nasıl Okunmalı?

Bu kitap yalnızca teknik bir programlama kitabı değildir. Aynı zamanda yeni bir düşünme biçimine giriş niteliği taşımaktadır. Kitap üç farklı düzeyde okunabilir.

İlk düzey kavramsal okumadır. Eğer okuyucu bilgi grafı yaklaşımının felsefi ve teorik yönüyle ilgileniyorsa; ağ bilimi, semantik web, ontolojiler ve yapay zekâ bölümleri temel bir çerçeve sunacaktır.

İkinci düzey teknik okumadır. Bu bölümlerde graf veri tabanları, Cypher sorguları, bilgi çıkarımı, GrafRAG sistemleri ve uygulama mimarileri ele alınacaktır.

Üçüncü düzey ise uygulama laboratuvarıdır. Özellikle Bölüm 9'da okuyucu; Python, Neo4j, Kùzu, embedding[4] modelleri ve local RAG sistemleri kullanarak kendi bilgi grafı projelerini geliştirebilecektir.

---

[2] SQL, Structured Query Language (Yapılandırılmış Sorgu Dili) ifadesinin kısaltmasıdır. Veritabanlarıyla konuşmak için kullanılan standart dildir. Basitçe söylemek gerekirse: SQL, bir veritabanına "ne istiyorsun?" demenin yoludur.

[3] JOIN, SQL'de farklı tablolardaki verileri ilişkiye göre birleştirme işlemidir.

[4] Bilişimde, yapay zekada ve özellikle doğal dil işlemede (NLP) embedding (gömme), kelimeler, cümleler, resimler veya kullanıcılar gibi karmaşık ve yüksek boyutlu verilerin, bilgisayarların kolayca işleyebileceği düşük boyutlu ve anlamlı sayısal vektörlere (matematiksel dizilere) dönüştürülmesi işlemidir.

Bu kitap boyunca temel amaç yalnızca “bir teknolojiyi öğretmek” değildir. Daha önemli olan, okuyucunun dünyaya ilişkiler üzerinden bakabilmesini sağlamaktır. Çünkü yapay zekâ çağında en değerli yeteneklerden biri, veriler arasındaki görünmeyen bağlantıları fark edebilme kapasitesidir.

BÖLÜM 1 — BİLGİ GRAFI NEDİR?

Bu bölümde bilgi grafı kavramının temel mantığını ve ortaya çıkış nedenlerini inceleyeceğiz. Günümüzde bilgi grafı yaklaşımı; arama motorlarından yapay zekâ sistemlerine, sosyal ağ analizlerinden bilimsel keşiflere kadar çok geniş bir kullanım alanına sahiptir. Ancak bu teknolojiyi anlamak için önce "veri", "enformasyon", "bilgi" ve "anlam" arasındaki farkları kavramak gerekir. Çünkü bilgi grafı yaklaşımı yalnızca veri depolamayı değil, veriler arasındaki ilişkilerden anlam üretmeyi hedefler.

Bölüm boyunca graf (çizge) düşüncesinin tarihsel gelişimini ele alacak; düğüm, bağlantı ve ilişki gibi temel kavramları açıklayacağız. Daha sonra ontoloji yaklaşımının neden modern bilgi sistemlerinin merkezine yerleştiğini göreceğiz. Semantik Web vizyonu, RDF yapısı[5], OWL dili[6] ve SPARQL sorgulama yaklaşımı[7] bu bağlamda değerlendirilecektir.

Bu yapı aslında 5 katmanda "zihinsel dönüşüm" gösterir:

1. 📥 Veri → Dünya

Ham veri henüz anlamlı değildir.

2. 🕸️ Veri → Graf

Veri artık ilişkilere dönüşür.

3. 🔗 Graf → RDF

Her ilişki "üçlü cümle" olur.

4. 🧩 RDF → OWL

Veriye **anlam ve kurallar** eklenir.

5. 🔍 OWL → Akıl Yürütme

---

[5] RDF (Resource Description Framework), veriyi "anlamlı ilişkiler" şeklinde temsil etmek için kullanılan bir semantik veri modelidir. En temel amacı, bilgiyi insanlar için okunabilir metinler yerine, makinelerin anlayabileceği ilişki ağı (graf) biçiminde ifade etmektir.

[6] OWL (Web Ontology Language), veriyi sadece "saklamak" için değil, ona anlam, kural ve mantık eklemek için kullanılan bir ontoloji dilidir. RDF'in üzerine inşa edilir ve "bilgi grafını akıllı hale getiren katman" gibi düşünülebilir.

[7] SPARQL (SPARQL Protocol and RDF Query Language), RDF formatında tutulan verileri sorgulamak için kullanılan standart sorgu dilidir. Yani SQL'in tablo dünyasındaki karşılığı neyse, SPARQL de graf ve semantik web dünyasının sorgu dilidir.

Sistem artık **çıkarım yapabilir**.

6. 🧠 Akıl Yürütme → YZ

LLM ve YZ ajan sistemleri bu bilgiyi kullanır. Bu mimarinin özü şudur:

RDF veri üretir
OWL anlam üretir
Graf yapı üretir
Akıl yürütme bilgi üretir
Büyük Dil Modelleri (LLM) karar üretir

OWL + RDF + Bilgi Grafı birleşimi, veriyi ham durumdan çıkarıp "akıl yürütülebilir bilgi sistemine" dönüştüren katmanlı bir bilişsel mimaridir.

Ayrıca bu bölümde bilgi grafı ile graf veri tabanı arasındaki farklar incelenecek; Neo4j, TigerGraf ve Amazon Neptune gibi modern platformların temel özellikleri tanıtılacaktır. Bölümün sonunda ise arama motorları, öneri sistemleri ve sosyal medya gibi günlük hayattan örneklerle bilgi grafı yaklaşımının gerçek dünyadaki etkisini göreceğiz.

Shannon, C. E. (1948). *A Mathematical Theory of Communication*. Bell System Technical Journal.

Wiener, N. (1948). *Cybernetics: Or Control and Communication in the Animal and the Machine*. MIT Press.

Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach*. Pearson.

Sowa, J. F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole.

1.1 Veri, Enformasyon, Bilgi ve Anlam

Modern dünyanın en çok kullanılan sözcüklerinden biri "veri"dir. Şirketler veriden söz eder, devletler veriyi toplar, üniversiteler veri üretir ve yapay zekâ sistemleri devasa veri kümeleriyle eğitilir. Ancak çoğu zaman veri, enformasyon, bilgi ve anlam kavramları birbirine karıştırılır. Oysa bilgi grafı yaklaşımını gerçekten anlayabilmek için önce bu kavramlar arasındaki farkı net biçimde görmek gerekir. Çünkü bilgi grafı sistemleri yalnızca veri depolamaz; veriler arasındaki ilişkilerden anlam üretmeye çalışır.

Veri: Ham Gerçeklik

Veri en temel düzeydir. İşlenmemiş gözlemler, ölçümler, kayıtlar ve semboller veri olarak adlandırılır. Tek başına çoğu zaman bağlam içermez.

Örneğin:

- 39.9

- 2026
- Bursa
- “Ali”
- 1453

bunların her biri birer veridir. Ancak bu değerlerin neyi temsil ettiğini bilmeden anlam üretmek mümkün değildir.

Bir sensörün ölçtüğü sıcaklık değeri, bir bankanın müşteri numarası, bir GPS koordinatı veya bir internet tıklaması veri olabilir. Veri hamdır. Yorumlanmamıştır. Çoğu zaman parçalıdır. Bilgisayar bilimlerinin ilk dönemlerinde temel amaç veriyi saklamaktı. Veri tabanları bu nedenle ortaya çıktı. Çünkü şirketler ve kurumlar giderek büyüyen veri yığınlarını düzenli biçimde depolamak zorundaydı. Ancak kısa sürede fark edildi ki yalnızca veri toplamak yeterli değildir.

Enformasyon: Verinin Düzenlenmiş Hali

Veri belirli bir bağlam içine yerleştirildiğinde enformasyona dönüşür.

Örneğin:

- “Bursa’da sıcaklık 39.9°C ölçüldü.”
- “1453 yılında İstanbul fethedildi.”
- “Ali son üç ayda beş ürün satın aldı.”

Artık elimizde yalnızca sayı veya sembol değil, yorumlanabilir bir içerik vardır. Enformasyon, verinin organize edilmiş biçimidir. Bir tablo, rapor, graf veya istatistiksel özet aslında enformasyon üretir. Geleneksel veri analitiğinin büyük bölümü bu düzeyde çalışır. İlişkisel veri tabanları da esas olarak enformasyon yönetimi sistemleridir. Tablolar, satırlar ve sütunlar aracılığıyla veriler düzenlenir ve erişilebilir hale getirilir. Fakat burada önemli bir sınır vardır: Enformasyon her zaman bilgi anlamına gelmez. Bir şirkette milyonlarca müşteri kaydı bulunabilir. Ancak müşteriler arasındaki davranış örüntüleri, sosyal etkiler veya gizli bağlantılar keşfedilmedikçe sistem hâlâ sınırlı bir anlayış düzeyindedir.

Bilgi: İlişkilerin Ortaya Çıkışı

Bilgi, enformasyonun yorumlanması ve ilişkiler içinde anlam kazanmasıyla ortaya çıkar. Örneğin bir e-ticaret sistemi şunu fark edebilir:

- Aynı ürünleri satın alan kullanıcılar benzer davranışlar göstermektedir.
- Belirli müşteriler aynı sosyal ağ içinde yer almaktadır.
- Bazı satın alma örüntüleri dolandırıcılık riski taşımaktadır.

Burada artık yalnızca kayıtlar değil, ilişkiler önemlidir. Bilgi grafı yaklaşımının temelinde de bu fikir vardır: Bilgi, nesnelerin kendisinden çok nesneler arasındaki bağlantılarda saklıdır. Aslında insan düşüncesi de büyük ölçüde bu şekilde çalışır. İnsan zihni bilgileri tek tek kutular halinde depolamaz. Kavramlar arasında bağlar kurar. Örneğin "Einstein" dediğimizde zihnimizde aynı anda şunlar canlanabilir:

- görelilik kuramı
- fizik
- Nobel Ödülü
- Almanya
- atom çağı
- bilim tarihi

Bu çağrışım zinciri ilişkisel bir yapıdır. İnsan belleği adeta biyolojik bir bilgi grafı gibi çalışır.

Anlam: Bağlamın Derinleşmesi

Anlam, yalnızca ilişki kurmakla değil, bu ilişkilerin bağlamını anlayabilmekle ortaya çıkar. Örneğin bir sistem şu cümleyi görebilir:

"Jaguar hızla ilerliyordu."

Buradaki "Jaguar" bir hayvan mı, bir otomobil markası mı, yoksa bir spor takımı mı?

Bu sorunun cevabı bağlama bağlıdır.

Anlam üretimi, modern yapay zekânın en zor problemlerinden biridir. Çünkü gerçek dünya belirsizlik, çok anlamlılık ve bağlamsallık içerir. Bilgi grafı sistemleri bu problemi azaltmaya çalışır. Çünkü bilgi graflarında varlıklar yalnızca kelime olarak değil, belirli ilişkiler ağı içindeki "şeyler" olarak temsil edilir.

Örneğin:

- Jaguar → hayvan → memeli → orman ekosistemi
- Jaguar → otomobil markası → İngiltere → lüks araç segmenti

Böylece sistem bağlama göre doğru anlamı seçebilir.

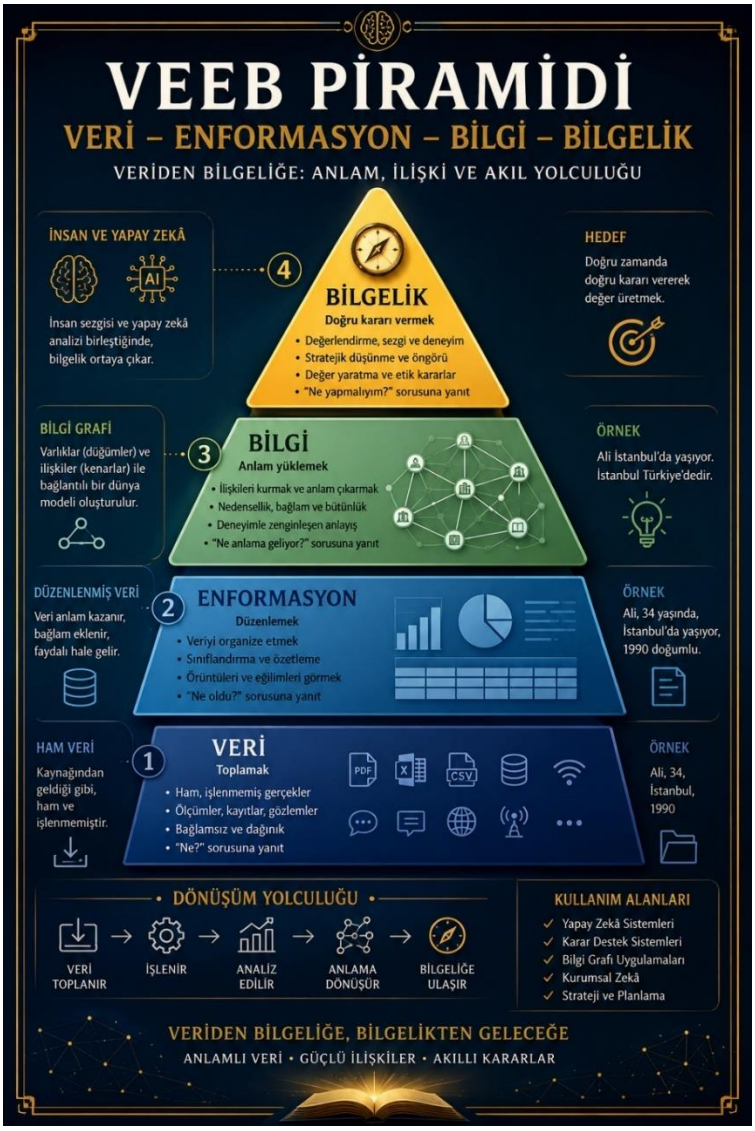
Bu yaklaşım özellikle arama motorları ve büyük dil modelleri açısından kritik hale gelmiştir. Çünkü modern yapay zekâ sistemlerinin en büyük problemlerinden biri, kelimeleri istatistiksel olarak işleyebilmesine rağmen bazen gerçek anlam ilişkilerini kaybetmesidir.

VEBB Piramidi

Bilgi yönetimi literatüründe sık kullanılan modellerden biri VEBB piramididir:

- Data (Veri)
- Information (Enformasyon)
- Knowledge (Bilgi)
- Wisdom (Bilgelik)

Şekil 1. VEEB Piramidi



Şekil 1'deki piramit, tabandan tavana doğru ham verinin nasıl işlenip anlam kazandığını ve en tepe noktada nasıl stratejik bir akla (bilgeliğe) dönüştüğünü basamak basamak gösterir. Her basamakta kavramların tanımı, yanıtladığı temel sorular ("Ne?", "Ne oldu?", "Ne anlama geliyor?", "Ne yapmalıyım?") ve süreci somutlaştıran birer örnek senaryo (Ali'nin yaş, şehir ve doğum yılı bilgileri üzerinden) yer almaktadır. Ayrıca sol taraftaki yan panellerde bu aşamaların graf veri tabanları (Bilgi Grafiği) ve insan-yapay zekâ entegrasyonu ile olan ilişkisi kurulmuştur.

Görselin alt bölümünde ise piramidin içindeki bu dikey tırmanış, **"Dönüşüm Yolculuğu"** başlığı altında yatay bir akış şemasıyla desteklenmiştir; süreç verinin toplanmasıyla başlar, işlenme,

analiz ve anlama evrelerinden geçerek bilgeliğe ulaşır. Sağ alt köşede ise bu modelin yapay zekâ sistemleri, karar destek mekanizmaları ve kurumsal zekâ gibi hangi kritik kullanım alanlarında karşılık bulduğu listelenmiştir. Genel hatlarıyla infografik, "Veriden bilgeliğe, bilgelikten geleceğe" mottosuyla, ham datayı doğru kararlara ve değere dönüştürmenin sistematik yol haritasını sunmaktadır.

Bu modele göre veri en alt düzeydir. Veri işlenerek enformasyona dönüşür. Enformasyon yorumlanınca bilgi oluşur. Bilgelik ise bilgiyi doğru bağlamda kullanabilme kapasitesidir. Bilgi grafı yaklaşımı özellikle veri ile bilgi arasındaki dönüşüm sürecinde kritik rol oynar. Çünkü bu sistemler ilişkileri görünür hale getirir. Aslında modern çağın temel problemi veri eksikliği değildir. Tam tersine, insanlık tarihin en büyük veri patlamasını yaşamaktadır. Sorun, bu devasa veri okyanusundan anlam çıkarabilmektir.

> Bu nedenle günümüzün en değerli sistemleri yalnızca veri depolayan sistemler değil; ilişkileri keşfeden, bağlam kuran ve anlam üreten sistemlerdir.

Neden Bilgi Grafı?

Bilgi grafı yaklaşımının yükselişi tam da bu ihtiyaçtan kaynaklanmaktadır. Çünkü klasik veri tabanı mantığında ilişkiler çoğu zaman ikincil düzeydedir. Oysa bilgi grafı yaklaşımında ilişkiler merkezdedir. Bir bilgi grafında:

- insanlar şirketlerle,
- şirketler teknolojilerle,
- teknolojiler patentlerle,
- patentler bilim insanlarıyla,
- bilim insanları üniversitelerle

bağlantılı hale gelir.

Böylece sistem yalnızca veri saklamaz; bir anlam haritası oluşturur. Bugün Google arama sonuçlarını geliştirmek için, Amazon ürün önerileri oluşturmak için ve Palantir operasyonel karar sistemleri kurmak için bilgi grafı yaklaşımını kullanmaktadır.

Yapay zekâ çağında bilgi artık statik bir arşiv değil; sürekli güncellenen, ilişkilerle genişleyen dinamik bir ağdır. Ve bu dönüşüm, bilgisayar bilimlerinin en önemli paradigma değişimlerinden birini temsil etmektedir.

Ackoff, R. L. (1989). *From Data to Wisdom*. Journal of Applied Systems Analysis.

Davenport, T. H., & Prusak, L. (1998). *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press.

 Zins, C. (2007). Conceptual Approaches for Defining Data, Information, and Knowledge. *Journal of the American Society for Information Science and Technology*.

1.2 Graf Düşüncesinin Tarihi

İnsanlık uzun süre dünyayı nesneler üzerinden anlamaya çalıştı. Ancak zamanla fark edildi ki dünyayı gerçekten biçimlendiren şey yalnızca nesnelerin kendisi değil, nesneler arasındaki ilişkilerdir. Bir insanı toplumdan, bir şirketi tedarik zincirinden, bir hücreyi biyolojik ağlardan veya bir fikri tarihsel bağlamından ayırdığımızda geriye eksik bir resim kalır.

Bugün "graf" ya da "ağ" düşüncesi olarak adlandırdığımız yaklaşımın kökleri aslında oldukça eskidir. Bilgi grafi sistemleri modern bilgisayar çağının ürünü gibi görünse de altında yatan düşünce biçimi yüzyıllar boyunca farklı alanlarda gelişmiştir.

Haritalardan Ağlara

İnsan zihni çok eski dönemlerden beri bağlantıları görselleştirmeye çalışıyordu. İlk ticaret yolları, soy ağaçları, yıldız haritaları ve şehir planları aslında ilkel ağ modelleriydi. İnsanlar noktalar ve bağlantılar üzerinden karmaşık sistemleri anlamlandırmaya çalışıyordu.

Özellikle denizcilik çağında haritalar yalnızca coğrafyayı değil, ilişkileri temsil etmeye başladı. Limanlar arasındaki ticaret yolları, askeri bağlantılar ve ekonomik akışlar bir tür ağ mantığı oluşturuyordu. Ancak graf düşüncesinin matematiksel temelleri 18. yüzyılda atıldı.

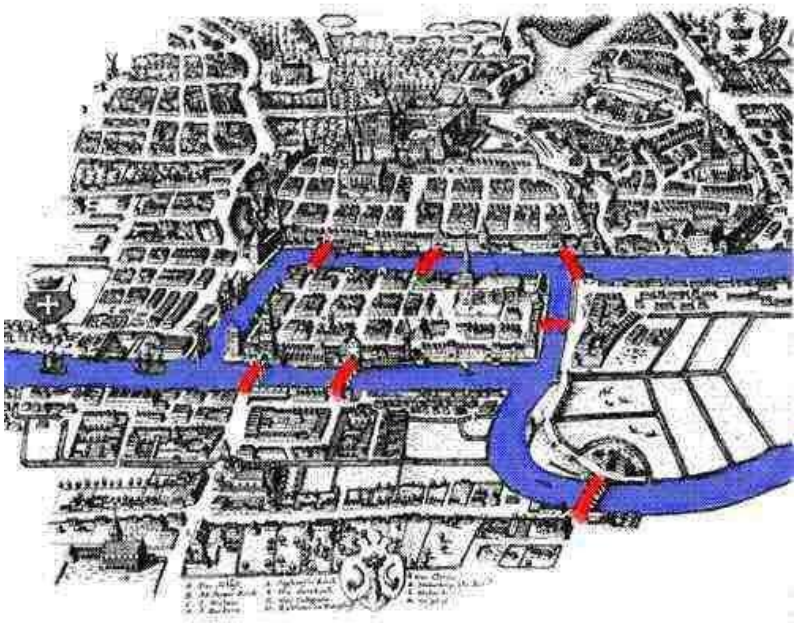
Euler ve Königsberg Köprüleri

Graf teorisinin başlangıcı genellikle İsviçreli matematikçi Leonhard Euler'in 1736 yılında çözdüğü ünlü "Königsberg Köprüleri Problemi"ne dayandırılır. O dönemde Königsberg şehri (bugünkü Kaliningrad), nehir ve adalarla bölünmüş bir yerleşimdi. Şehirdeki insanlar şu soruyu tartışıyordu: "Şehirdeki yedi köprünün her birinden yalnızca bir kez geçerek yürümek mümkün mü?"

Bu problem ilk bakışta eğlenceli bir bulmaca gibi görünüyordu. Ancak Euler problemi tamamen farklı biçimde ele aldı. Köprülerin fiziksel uzunlukları, mimarileri veya estetik özellikleriyle ilgilenmedi. Bunun yerine yalnızca bağlantı yapısına odaklandı:

- Kara parçaları → düğümler (node)
- Köprüler → bağlantılar (edge)

Şekil 2. Königsberg Köprüleri



Böylece modern graf teorisinin temel fikri ortaya çıktı: Bazı problemler nesnelerin kendisinden çok ilişkileri üzerinden anlaşılabilir. Euler, bu yürüyüşün mümkün olmadığını matematiksel olarak kanıtladı. Ancak daha da önemlisi, yeni bir düşünme biçiminin kapısını açtı. Bugün bilgi grafi sistemlerinin temelinde hâlâ aynı mantık vardır: Önemli olan yalnızca verinin kendisi değil, veriler arasındaki bağlantı yapısıdır.

19. Yüzyıl: Ağların Bilimselleşmesi

19.yüzyılda graf düşüncesi farklı alanlara yayılmaya başladı. Demiryolu ağları, telgraf sistemleri ve elektrik şebekeleri büyüdükçe mühendisler karmaşık bağlantıları analiz etmek zorunda kaldılar. Endüstri Devrimi aslında büyük ölçüde ağların genişleme süreciydi. Aynı dönemde sosyoloji de ilişkisel düşünceyi geliştirmeye başladı. İnsan topluluklarının yalnızca bireylerden oluşmadığı; bireyler arasındaki bağlantıların toplumsal yapıyı belirlediği fark edildi.

Ekonomi dünyasında ise ticaret ağları, finansal bağlantılar ve küresel akış sistemleri önem kazandı. Böylece graf (çizge) yaklaşımı yalnızca matematiksel bir araç olmaktan çıkıp dünyayı anlamanın genel bir yöntemi haline gelmeye başladı.

20. Yüzyıl: Ağ Biliminin Doğuşu

20.yüzyılda graf teorisi büyük bir hızla gelişti. Özellikle bilgisayar bilimlerinin ortaya çıkışı bu süreci hızlandırdı. Telefon ağları, ulaşım sistemleri ve internetin öncülleri; karmaşık bağlantı yapılarını modellemeyi zorunlu hale getirdi. 1940'lardan itibaren bilgisayar bilimcileri şu ağlarla ilgili şu sorularla ilgilenmeye başladı:

- En kısa yol nasıl bulunur?
- Ağ içinde bilgi nasıl yayılır?

- Hangi düğümler daha merkezi konumdadır?
- Ağ çökerse sistem nasıl etkilenir?

Bu sorular bugün sosyal ağ analizlerinden lojistik optimizasyonuna kadar pek çok alanda kullanılmaktadır. Aynı dönemde psikoloji ve nörobilim de insan beyninin ağ yapısına benzediğini göstermeye başladı. Beyin, milyarlarca nöron arasındaki bağlantılar üzerinden çalışıyordu. Böylece graf düşüncesi biyolojiye de girmiş oldu.

İnternet ve Ağ Patlaması

1990'lı yıllarda internetin yaygınlaşması graf düşüncesini yeni bir aşamaya taşıdı. World Wide Web aslında devasa bir bağlantılar sistemiydi. Web sayfaları birbirine hyperlink'lerle bağlanıyordu. Bu yapı doğal olarak bir graf oluşturuyordu. İnternet büyüdükçe yeni problemler ortaya çıktı:

- Hangi sayfalar daha önemli?
- Bilgi ağ içinde nasıl yayılıyor?
- Sahte bağlantılar nasıl tespit edilir?
- İnsanlar internette nasıl kümeleniyor?

Bu soruların cevabı graf teorisi içinde bulundu.

Örneğin Google'ın ilk büyük başarısı olan PageRank algoritması tamamen ağ mantığına dayanıyordu. Google yalnızca kelimelere bakmıyor, hangi sayfanın hangi sayfalar tarafından referans verildiğini analiz ediyordu. Yani web'i bir graf olarak görüyordu. Bu yaklaşım modern bilgi çağının en büyük dönüşümlerinden birini başlattı.

Sosyal Ağlar ve İlişkisel Dünya

2000'li yıllarla birlikte sosyal medya platformları graf düşüncesini gündelik hayatın merkezine taşıdı. Facebook, LinkedIn ve diğer platformlar insan ilişkilerini doğrudan ağ yapıları üzerinden modelledi.

Artık:

- arkadaşlıklar,
- takip ilişkileri,
- mesajlaşmalar,
- ilgi alanları,
- topluluk yapıları

birer graf problemi haline gelmişti.

Modern sosyal medya aslında devasa bir bilgi grafı sistemidir. Bugün öneri motorlarından reklam sistemlerine kadar pek çok dijital mekanizma bu ilişkisel yapı üzerinde çalışmaktadır.

Semantik Web ve Bilgi Grafları

2000'li yılların en önemli fikirlerinden biri de Semantik Web yaklaşımı oldu. Tim Berners-Lee, internetin yalnızca insanlar tarafından okunabilen bir belge sistemi olmaktan çıkıp makineler tarafından anlaşılabilen bir bilgi ağına dönüşmesini hayal ediyordu. Bu yaklaşımın temelinde şu fikir vardı:

"Web yalnızca dokümanları değil, anlamları da bağlamalıdır."

Böylece:

- RDF,
- OWL,
- ontolojiler,
- semantik ilişkiler

gibi kavramlar ortaya çıktı.

Bilgi grafı yaklaşımı işte bu düşünsel zeminden doğdu.

Yapay Zekâ Çağında Graf Düşüncesi

Bugün graf yaklaşımı yeni bir döneme girmiş durumda. Modern yapay zekâ sistemleri çok büyük veri kümeleri üzerinde çalışıyor. Ancak yalnızca veri miktarı yeterli değil. Sistemlerin ilişkileri anlayabilmesi gerekiyor. Bu nedenle bilgi grafı sistemleri yeniden büyük önem kazandı.

Özellikle:

- GrafRAG,
- neuro-symbolic AI,
- graf neural networks,
- semantik ajan sistemleri

gibi alanlar graf düşüncesini yapay zekâ ile birleştirmektedir.

Şekil 3'de yer alan görsel, yapay zekâ teknolojilerinde graf (graph) yapılarının kullanımını ve bu yaklaşımların sağladığı avantajları özetleyen Türkçe bir infografiktir. Görselin merkezinde, etrafına ışık dalgaları ve veri hatları yayan mavi renkli, parıldayan bir insan beyni illüstrasyonu yer almaktadır. Bu merkezden dışarıya doğru uzanan dört ana akış hattı, graf tabanlı modern yapay zekâ yaklaşımlarını temsil eden dört renkli daireye bağlanmaktadır.

Şekil 3. Graf Düşüncesi ve Yapay Zekâ Birleşimi

Şekil 3'de yer alan dört ana yaklaşım ve açıklamaları şu şekildedir:

- **Graph Neural Networks (GNN) (Yeşil Daire):** Sol üstte yer alır. Graf yapısındaki verileri doğrudan işleyerek, düğüm (node) ve kenar (edge) özelliklerini öğrendiği belirtilmiştir. Kullanım alanları olarak sosyal ağ analizi, ilaç keşfi ve sahtekarlık tespiti sıralanmıştır. Bu daire, merkeze **"Bilgi Grafları"** köprüsüyle bağlanır.
- **Graph Retrieval-Augmented Generation (GraphRAG) (Mavi Daire):** Sağ üstte yer alır. Büyük dil modellerinin (LLM) bilgilerini, dışarıdaki zenginleştirilmiş graf veri tabanlarından alınan bağlamla birleştirdiğini ifade eder. Sağladığı fayda olarak daha doğru, bağlamsal ve güncel metin üretimi örnek gösterilmiştir. Bu daire de merkeze yine **"Bilgi Grafları"** hattıyla bağlıdır.
- **Neuro-Symbolic AI (Turuncu Daire):** Sol altta yer alır. Derin öğrenmenin veri odaklı gücünü, semantik mantık ve kural tabanlı sistemlerle birleştirdiği anlatılmaktadır. Şeffaf, açıklanabilir kararlar ve daha güçlü genelleme yeteneği sunduğu vurgulanmıştır. Merkeze **"Mantık"** ve **"Öğrenme"** kavramları üzerinden bağlanır.
- **Semantik Ajan Sistemleri (Kırmızı Daire):** Sağ altta yer alır. Anlamlı veri modelleri (ontolojiler) kullanarak, ajanların karmaşık görevleri planlamasını ve yürütmesini sağladığı aktarılmıştır. Örnek olarak akıllı iş akışları, otomatik problem çözme ve işbirlikçi sistemler verilmiştir. Merkeze **"Akıl Yürütme"** kavramıyla bağlanır.

Dünyayı Bir Ağ Olarak Görmek

Graf düşüncesinin en önemli özelliği dünyaya farklı bir gözle bakmayı öğretmesidir. Bu yaklaşımda hiçbir şey tamamen bağımsız değildir.

- Bilim insanları atıf ağlarının parçasıdır.

- Şirketler tedarik zincirlerine bağlıdır.
- İnsanlar sosyal ilişkiler içinde yaşar.
- Şehirler enerji ve ulaşım ağlarıyla çalışır.
- Yapay zekâ sistemleri bilgi ağları üzerinde gelişir.

Aslında modern uygarlık giderek daha fazla ilişkisel hale gelmektedir. Bilgi grafı yaklaşımı da tam olarak bu gerçeğin dijital temsilidir. Bu nedenle graf düşüncesi yalnızca teknik bir araç değil; aynı zamanda çağımızı anlamanın yeni yollarından biridir.

Euler, L. (1736). *Solutio problematis ad geometriam situs pertinentis* (Seven Bridges of Königsberg).

Harary, F. (1969). *Graph Theory*. Addison-Wesley.

Newman, M. (2010). *Networks: An Introduction*. Oxford University Press.

1.3 Düğüm (Node), Bağlantı (Edge), İlişki (Relation)

Bilgi grafı sistemlerinin temelinde oldukça basit ama çok güçlü bir fikir vardır: Dünya, birbirleriyle bağlantılı varlıklardan oluşur. İnsanlar, şehirler, şirketler, kitaplar, bilimsel keşifler, makineler ve olaylar; görünmez ilişkiler ağı içinde birbirine bağlıdır. Graf (graf) yaklaşımı bu karmaşık yapıyı üç temel bileşen üzerinden modellemeye çalışır:

- Düğüm (node)
- Bağlantı (edge)
- İlişki (relation)

Bu üç kavram bilgi grafı dünyasının alfabesi gibidir. Daha karmaşık tüm yapılar aslında bu temel yapı taşlarının birleşiminden oluşur.

Düğüm (Node): Varlığın Kendisi

Düğüm (node), graf içindeki temel varlığı temsil eder. Gerçek dünyada "şey" olarak düşündüğümüz hemen her unsur bir düğüm olabilir.

Örneğin:

- bir insan,
- bir şirket,
- bir şehir,
- bir film,

- bir makale,
- bir ürün,
- bir protein,
- bir web sayfası

birer düğüm olarak modellenebilir.

Bir bilgi grafı sisteminde her düğüm genellikle belirli özellikler (property) taşır. Örneğin bir "insan" düğümünde şu bilgiler bulunabilir:

- isim,
- yaş,
- doğum tarihi,
- meslek,
- ülke

gibi özellikler.

Örneğin:

Düğüm: Albert Einstein
Özellikler:

- meslek = fizikçi
- doğum yılı = 1879
- ülke = Almanya

Burada önemli nokta şudur: Düğüm yalnızca bir veri satırı değildir. O, sistem içinde anlam taşıyan bağımsız bir varlıktır.

İlişkisel veri tabanlarında bir kayıt çoğu zaman yalnızca tablo içindeki bir satırdır. Ancak bilgi grafı yaklaşımında düğüm daha zengin bir kimliğe sahiptir. Çünkü düğüm yalnızca kendi özellikleriyle değil, diğer düğümlerle kurduğu ilişkilerle de tanımlanır.

Aslında gerçek hayatta da durum böyledir. Bir insanı yalnızca adıyla değil; arkadaşları, çalıştığı kurum, yaşadığı şehir ve yaptığı çalışmalar üzerinden tanırız.

Bağlantı (Edge): Bağlantının Kendisi

Düğümler tek başına yeterli değildir. Graf sistemlerinin asıl gücü düğümler arasındaki bağlantılardan gelir. İşte bu bağlantılar bağlantı (edge) olarak adlandırılır.

Bir bağlantı, iki düğüm arasında bağ kurar.

Örneğin:

- Einstein → Princeton Üniversitesi'nde çalıştı
- Shakespeare → Hamlet'i yazdı
- Bursa → Türkiye'de bulunur
- Bir müşteri → bir ürünü satın aldı

Bu bağlantılar graf yapısının omurgasını oluşturur.

Klasik veri tabanı sistemlerinde ilişkiler çoğu zaman dolaylıdır. Birleştirme (join) işlemleriyle bağlantılar kurulmaya çalışılır. Ancak graf yaklaşımında ilişki sistemin merkezindedir.

Bu nedenle graf veri tabanlarında dolaşım (traversal) yani bağlantılar üzerinde gezinme işlemleri son derece önemlidir.

Örneğin bir sosyal medya sisteminde şu soru sorulabilir:

"Ali'nin arkadaşlarının arkadaşları arasında aynı üniversitede çalışan kaç kişi var?"

İlişkisel veri tabanlarında bu tür sorgular karmaşık hale gelirken graf yapılarında doğal biçimde modellenebilir.

Çünkü graf düşüncesi bağlantıları birinci sınıf unsur olarak kabul eder.

İlişki (Relation): Bağlantının Anlamı

Bağlantı (edge) fiziksel bağlantıyı temsil ederken ilişki (relation), o bağlantının anlamını ifade eder.

Örneğin iki insan arasındaki bağlantı farklı türlerde olabilir:

- arkadaşlık,
- akrabalık,
- iş ilişkisi,
- akademik işbirliği,
- takip ilişkisi

Bunların her biri farklı ilişki türüdür.

Aynı şekilde:

- bir bilim insanı → bir teori geliştirdi
- bir şirket → bir teknolojiyi satın aldı
- bir hasta → bir ilacı kullandı

ilişkileri farklı anlamsal katmanlar taşır.

Bilgi grafi yaklaşımını güçlü yapan şey tam da budur: Sistem yalnızca bağlantı kurmaz, bağlantının türünü de bilir.

Bu nedenle bilgi grafı sistemleri anlamsal (semantic) yapı taşır. Yani yalnızca veri değil, anlam da temsil edilir.

Gerçek Dünya Bir Graf mıdır?

Aslında modern dünyanın büyük bölümü doğal olarak graf yapısındadır.

Örneğin:

Sosyal Ağlar

İnsanlar birbirine arkadaşlık, takip veya iletişim ilişkileriyle bağlanır.

Ulaşım Sistemleri

Şehirler yollar, havayolları ve demiryollarıyla birbirine bağlanır.

İnternet

Web sayfaları köprü bağlantılarıyla (hyperlink) ilişkilidir.

Biyoloji

Proteinler karmaşık etkileşim ağları oluşturur.

Finans

Şirketler yatırım, ortaklık ve ödeme ilişkileriyle bağlanır.

Akademik Dünya

Makaleler birbirine atıf verir. Yani graf yaklaşımı yapay bir model değil, gerçek dünyanın ilişkisel doğasının dijital temsilidir.

Yönlü ve Yönsüz Grafler

Bazı ilişkiler yön içerir, bazıları içermez.

Örneğin:

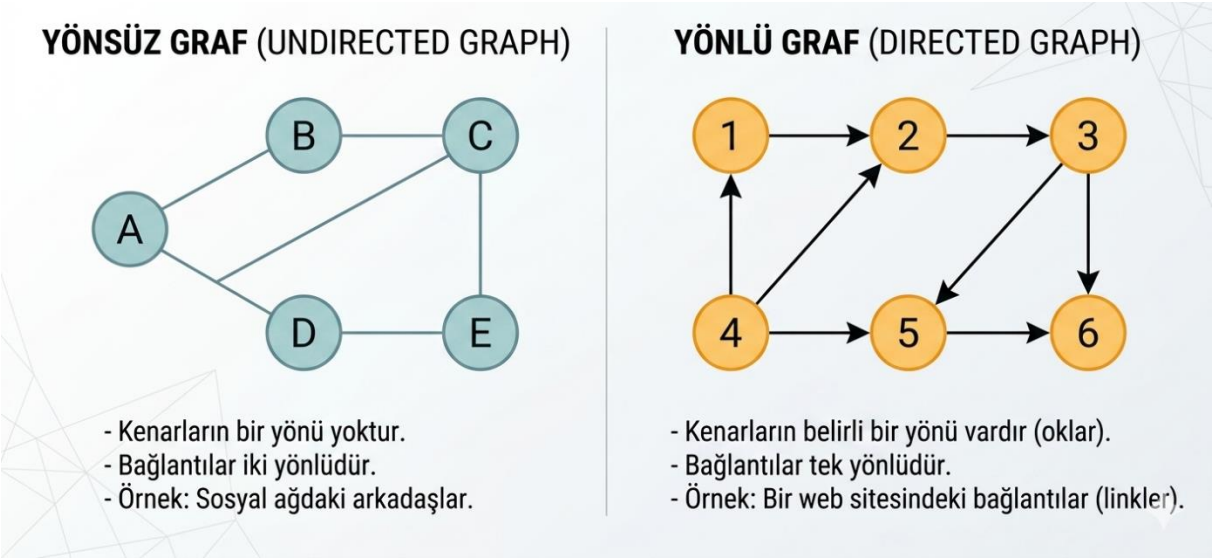
- “Ali, Ayşe’yi takip ediyor”
  ilişkisi yönlüdür.

Ama:

- “Ali ve Ayşe arkadaş”
  ilişkisi çift yönlü düşünülebilir.

Bu nedenle graf sistemlerinde iki temel yapı bulunur.

Şekil 4. Yönsüz ve Yönlü Graf Örnekleri



Yönlü Graf (Directed Graf)

Bağlantının yönü vardır.

Örneğin:

- sosyal medya takip ilişkileri,
- web bağlantıları,
- para transferleri

Yönsüz Graf (Undirected Graf)

Bağlantı karşılıklıdır.

Örneğin:

- arkadaşlık ilişkileri,
- ortak üyelikler

Gerçek dünya sistemlerinde çoğu graf aslında yönlüdür.

Ağırlıklı Bağlantı (Weighted Edge)

Bazı ilişkiler diğerlerinden daha güçlüdür.

Örneğin:

- bir kişiyle her gün görüşmek,
- yılda bir kez görüşmekten farklıdır.

Bu nedenle bağlantılara ağırlık (weight) eklenebilir.

Örneğin:

- iletişim sıklığı,
- ticaret hacmi,
- etkileşim yoğunluğu

gibi ölçüler bağlantı ağırlığı olarak tutulabilir.

Sosyal medya algoritmaları büyük ölçüde bu mantıkla çalışır. Sistem yalnızca bağlantı olup olmadığına değil, bağlantının gücüne de bakar.

Özellik Grafı Yaklaşımı (Property Graf Model)

Modern graf veri tabanlarının çoğu özellik grafiği (property graf) modelini kullanır. Bu modelde:

- düğümlerin özellikleri vardır,
- bağlantıların da özellikleri olabilir.

Şekil 5. Özellik Grafı

Örneğin:

Düğüm:

- kişi = Ali
- yaş = 42

Bağlantı:

- arkadaşlık süresi = 15 yıl
- iletişim sıklığı = yüksek

Bu yapı bilgi grafı sistemlerini son derece esnek hale getirir. Özellikle Neo4j gibi platformlar özellik grafiği yaklaşımını yaygınlaştırmıştır.

Şekil 5, modern graf veritabanlarının temel yapısı olan **Özellik Grafiği (Property Graph)** modelini dört temel düğüm üzerinden somutlaştıran, karanlık modda tasarlanmış estetik bir infografiktir. Görselin tam merkezinde, graf yapılarının akıllı veri bağlamını simgeleyen parıldayan bir insan beyni illüstrasyonu yer alır ve etrafı dairesel veri hatlarıyla çevrelenmiştir. Bu merkezi yapının dört köşesine ise şeffaf, köşeleri yuvarlatılmış kare paneller halinde yerleştirilmiş **Müşteri, Ürün, Arkadaş ve Kategori** etiketli dört ana düğüm (node) konumlandırılmıştır. Her bir düğüm paneli; kendi kimlik numarasını (id), ismini, türünü veya konumunu barındıran anahtar-değer çifti şeklindeki özellik (property) kutucuklarıyla desteklenmiştir.

Düğümler arasındaki veri akışı ve semantik bağlar ise yönlü, renkli oklar ve bu okların üzerine yazılmış ilişki etiketleriyle (relationships) açıkça gösterilmiştir. Örneğin, sol üstteki **Müşteri** düğümünden sağ üstteki **Ürün** düğümüne doğru SATIN_ALDI etiketli bir ok uzanmakta ve bu ilişkinin üzerinde alışverişin yapıldığı "tarih" bilgisi bir özellik olarak yer almaktadır. Benzer şekilde, Müşteri'den sol alttaki **Arkadaş** düğümüne ÖNERDİ, Arkadaş'tan sağ alttaki **Kategori** düğümüne İNCELEDİ ve Ürün'den Kategori'ye doğru AİTTİR yönlü ilişkileri uzanmaktadır. İlişkilerin de tıpkı düğümler gibi kendi üzerlerinde "puan", "yorum" veya "tarih" gibi spesifik nitelikler taşıyabildiğini net bir biçimde gösteren bu görsel, en altta yer alan *"Veriler, ilişkiler ve özelliklerle anlamlı graflar"* mottosuyla mizanpajı tamamlamaktadır.

Gezinme: Ağ İçinde Yolculuk

Graf sistemlerinin en kritik ve ayırt edici özelliklerinden biri, veri tabanındaki bağımsız tabloları birleştirmeye (JOIN işlemlerine) ihtiyaç duymadan, doğrudan veri yolları üzerinde akıcı bir şekilde hareket edebilmeyi sağlayan gezinme (traversal) yeteneğidir. Dolaşım, özü itibarıyla graf evreni içerisindeki düğümler (varlıklar) arasında, onları birbirine bağlayan kenarlar (ilişkiler) boyunca adım adım ilerleme ve bu hatları takip ederek yeni rotalar çizme işlemidir.

Gerçek hayattan ve veri modellerinden örnekler vermek gerekirse:

- Bir kişinin doğrudan bağlı olduğu arkadaşlarına gitmek (1. derece ilişkiler),
- O arkadaşların da kendi çevrelerini ve arkadaşlarını bulmak (2. derece ilişkiler),
- Farklı bireyler arasındaki gizli veya açık ortak bağlantıları, kesişim kümelerini keşfetmek tamamen birer dolaşım işlemidir.

Günümüzde veri yoğun çalışan modern mimariler, milyarlarca veri noktası arasındaki örüntüleri yakalamak için bu dolaşım mantığına sırtını yaslar:

- Modern Öneri Sistemleri (Recommendation Systems): *"Bu ürünü alanlar, şu ürünü de aldı"* ya da *"Arkadaşınızın beğendiği bu içerik sizin de ilginizi çekebilir"* tarzındaki algoritmalar, graf üzerindeki komşuluk ilişkilerini anlık olarak tarayarak kişiselleştirilmiş tavsiyeler üretir.
- Dolandırıcılık Analizleri (Fraud Detection): Finansal suçları ve organize dolandırıcılık şebekelerini tespit etmek için banka hesapları, IP adresleri, telefon numaraları ve para transferleri arasındaki şüpheli halkalar ile dolaylı bağlar gezinme algoritmalarıyla saniyeler içinde ortaya çıkarılır.
- Bilgi Keşif Araçları (Knowledge Discovery): Yapılandırılmamış devasa veri yığınları içinden anlamlı semantik ilişkiler çıkarmak ve büyük dil modellerini zenginleştiren bilgi graflarında (Knowledge Graphs) kaybolmadan doğru bağlamı yakalamak yine bu ağ içi yolculuklar sayesinde mümkün olur.

Aslında graf sistemlerindeki bu teknik dolaşım mantığı, yapay bir icattan ziyade doğanın ve insan zihninin çalışma prensibinin dijital dünyadaki kusursuz bir taklididir. İnsan beni de bilgiyi izole klasörlerde değil, muazzam bir nöral ağ içinde birbirine bağlı kavramlar halinde saklar.

Düşünce dünyamız tam anlamıyla bir graf dolaşımıdır:

- Bir andan diğerine geçerken çizgisel bir sıra takip etmeyiz; bir düşünce, aradaki anlamsal bir bağ vasıtasıyla tamamen başka bir düşünceyi tetikler.
- Bir kavram, hafızamızdaki onlarca farklı anıyı, mekanı veya kokuyu aynı anda çağrıştırır.

Örneğin "Bursa" kelimesi zihnimizde doğrudan "İskender Kebap" düğümüne bağlanabilir, oradan "Yeşil Türbe"ye, oradan da çocuklukta yapılan bir aile seyahatinin anısına uzanabilir. İşte insan beyninin ürettiği bu zihinsel çağrışım zincirleri, doğanın bize sunduğu en mükemmel ve en dinamik biyolojik dolaşım (traversal) örnekleridir. Yapay zekâ ve graf veritabanları da veriyi tam olarak insanın düşündüğü bu doğal formda işleyerek geleceğin akıllı sistemlerini inşa etmektedir.

Düğümler ve İlişkiler Dünyası: Ağ Gözlüğüyle Evrene Bakmak

Graf yaklaşımının insan düşüncesine ve veri bilimine sunduğu en köklü katkı, sadece teknik bir veritabanı modeli olması değil; bizlere **dünyayı ve olguları tamamen farklı bir gözlükle görmeyi öğretmesidir**. Geleneksel yaklaşımlar varlıkları izole kutular, satırlar veya kategoriler

halinde tekil olarak ele alırken; graf düşüncesi her varlığı evrensel bir dokunun kesişim noktası olarak kabul eder. Bu felsefi ve teknik dönüşüm sayesinde:

- **İnsanlar yalnızca birer birey değildir:** Her insan; ailesi, arkadaşları, tükettiği içerikler, seyahat ettiği rotalar ve yaptığı finansal işlemlerle sürekli genişleyen bir sosyal ve operasyonel ağın merkezindeki dinamik bir düğümdür (node).
- **Şirketler yalnızca hukuki kurumlar değildir:** Her şirket; tedarikçileri, alt yüklenicileri, rakipleri, yönetim kurulu bağları, çalışanları ve müşteri etkileşimleriyle yaşayan, organik ve devasa bir ekonomik ekosistemin parçasıdır.
- **Bilim yalnızca statik makalelerden oluşmaz:** Bilim; yazarlar, atıflar, laboratuvarlar, deney verileri, formüller ve birbirini fonlayan projeler arasındaki entelektüel köprülerle örülmüş küresel bir yaşayan bilgi ağacıdır.

Kısacası graf düşüncesinde hiçbir şey tek başına var olmaz; **her şey, kendisini tanımlayan ve var eden bütünsel bir ilişki ağının parçasıdır**.

Bilgi grafı sistemleri, gerçek dünyadaki bu karmaşık, dinamik ve çok boyutlu ilişkisel evreni dijital dünyada aslına en uygun biçimde temsil etme çabasından doğmuştur. Veriyi tablolara hapsetmek yerine, anlamlı bağlar (semantik ilişkiler) ve ontolojiler yardımıyla yaşayan bir haritaya dönüştürür. Bugün küresel teknoloji devleri ve stratejik analitik platformları, rekabet avantajlarını ve operasyonel güçlerini büyük ölçüde bu mantığa borçludur:

- **Google:** Arama motoru olmanın ötesine geçerek geliştirdiği *"Bilgi Grafı"* altyapısı sayesinde milyarlarca kavramı, tarihi kişiyi, mekânı ve olguyu birbirine bağlar. Bu sayede arama çubuğuna bir kelime yazdığınızda Google size sadece web sitelerini listelemez; aradığınız şeyin ne olduğunu "anlayarak" yan panelde ilişkili tüm bilgileri (doğum tarihi, eserleri, çağdaşları vb.) bir bütün olarak sunar.
- **Amazon:** Dünyanın en gelişmiş öneri sistemlerini (Recommendation Systems) bu hiyerarşik olmayan esnek yapıyla kurgular. Sizin satın alma geçmişiniz, sepetiniz, benzer profildeki diğer kullanıcıların klik hareketleri ve ürün kategorileri graf üzerinde anlık olarak taranarak, nokta atışı ve kişiselleştirilmiş tüketici yönlendirmeleri yapılır.
- **Palantir:** Özellikle savunma, istihbarat, büyük finans ve operasyonel analiz altyapılarında graf mantığını en uç noktada kullanır. Milyonlarca farklı kaynaktan gelen uçuş bilgileri, banka transferleri, telefon sinyalleri ve şirket ortaklıkları gibi dağınık verileri bir bilgi grafında birleştirerek organize suç yapılarını veya karmaşık tedarik zinciri krizlerini saniyeler içinde analiz eder.

Dijital dönüşümün ilk evrelerinde en büyük meydan okuma veriyi toplamak ve depolamaktı. Ancak günümüzde veri gölleri ve sunucular zaten taşma noktasına gelmiştir. Bu yüzden modern çağın en kritik ve hayati sorusu artık yalnızca **“Hangi veri var?”** sorusu değildir. Bu soru, eski dünyanın izole veri havuzlarında kalmıştır.

Bugünün ve geleceğin asıl stratejik sorusu şudur: “Bu veriler birbirine nasıl bağlı?” Çünkü ham verinin kendi başına taşıdığı değer sınırlıdır; asıl zekâ, bağlam ve büyük resim, o verilerin arasındaki gizli veya açık bağlantılarda saklıdır. Dünyayı düğümler ve ilişkiler üzerinden okumayı başaran sistemler, veriden sadece enformasyon değil, geleceği öngören köklü bir "bilgelik" üretirler.

Barabási, A.-L. (2016). *Network Science*. Cambridge University Press.

Easley, D., & Kleinberg, J. (2010). *Networks, Crowds, and Markets*. Cambridge University Press.

1.4 Ontoloji Nedir?

Bilgi grafı dünyasına girildiğinde en sık karşılaşılan ve en kritik kavramlardan biri “ontoloji”dir. Ontoloji, ilk bakışta felsefi bir terim gibi görünür ve aslında kökeni de felsefeye dayanır. Ancak modern bilgi sistemlerinde ontoloji, çok daha pratik ve teknik bir anlam kazanmıştır: Ontoloji bir alanın varlıklarını (entity), bu varlıkların türlerini ve aralarındaki ilişkileri tanımlayan yapısal bir modeldir. Basit bir ifadeyle ontoloji, “bir dünyayı nasıl modelleyeceğimizi belirleyen şema”dır.

> Ontoloji bir alanın varlıklarını (entity), bu varlıkların türlerini ve aralarındaki ilişkileri tanımlayan yapısal bir modeldir.

Ontolojinin Felsefi Kökeni

Ontoloji kavramı, insan düşüncesinin en eski ve en derin sorularından birine dayanır: “Gerçeklik nedir?” Antik Yunan filozofları için ontoloji, yalnızca nesneleri saymak değil, varlığın doğasını anlamaya çalışmak anlamına geliyordu. Özellikle Aristoteles, var olan her şeyin belirli kategoriler altında incelenebileceğini savunmuş ve böylece sınıflandırma düşüncesinin temellerini atmıştır. Ona göre dünya; nesneler, özellikler, nedenler ve ilişkiler ağı içinde anlaşılabilirdi. Bu yaklaşım, yalnızca felsefeyi değil, modern bilimin düşünsel mimarisini de derinden etkiledi.

Aristoteles’in “kategoriler” yaklaşımı, ontolojinin daha sonraki yüzyıllarda sistematik bir düşünce alanına dönüşmesini sağladı. Orta Çağ filozofları Tanrı, insan ve doğa arasındaki ilişkileri ontolojik çerçevede açıklamaya çalışırken; modern dönemde René Descartes, Immanuel Kant ve Martin Heidegger gibi düşünürler varlığın anlamı üzerine yeni tartışmalar geliştirdiler. Böylece ontoloji, yalnızca “ne vardır?” sorusuyla sınırlı kalmadı; “bir şeyi var olarak kabul etmenin koşulları nelerdir?” sorusunu da içeren daha geniş bir düşünce alanına dönüştü.

Bu felsefi arka plan, günümüzde bilgi teknolojilerinde kullanılan ontoloji kavramının da temelini oluşturur. Çünkü bilgisayar sistemleri yalnızca veri depolamakla yetinmez; veriler arasındaki ilişkileri anlamlandırmaya da ihtiyaç duyar. İnsan zihni için oldukça doğal olan kavramsal

bağlantılar, bilgisayar açısından açık biçimde tanımlanmadığında belirsizlik ortaya çıkar. Örneğin bir insan için “doktor”, “hastane” ve “tedavi” kavramları arasında güçlü ilişkiler vardır. Ancak bilgisayar bu ilişkileri kendiliğinden kuramaz. İşte ontoloji burada devreye girer: kavramların, özelliklerin ve ilişkilerin sistematik biçimde tanımlandığı bir bilgi haritası oluşturur.

Bilgisayar bilimlerinde ontoloji, belirli bir alanın kavramsal modelini ifade eder. Bir başka deyişle, o alan içindeki nesnelerin hangi sınıflara ait olduğunu, hangi özellikleri taşıdığını ve birbirleriyle nasıl ilişkiler kurduğunu tanımlar. Bu nedenle modern ontolojiler, yalnızca sözlük benzeri tanımlar içermez; aynı zamanda mantıksal kurallar ve çıkarım mekanizmaları da barındırır. Böylece makineler yalnızca veriyi saklamaz, aynı zamanda veriler arasında anlamlı bağlantılar kurabilir.

Özellikle bilgi grafı sistemleri, semantik web teknolojileri ve yapay zekâ uygulamaları ontolojik düşünceden yoğun biçimde yararlanır. Günümüzde Google, IBM ve Palantir Technologies gibi şirketler büyük veri kümelerini anlamlandırmak için ontoloji tabanlı sistemler geliştirmektedir. Çünkü modern dünyada sorun artık yalnızca veri eksikliği değil; aşırı veri içinde anlamlı ilişkileri keşfedebilme problemidir.

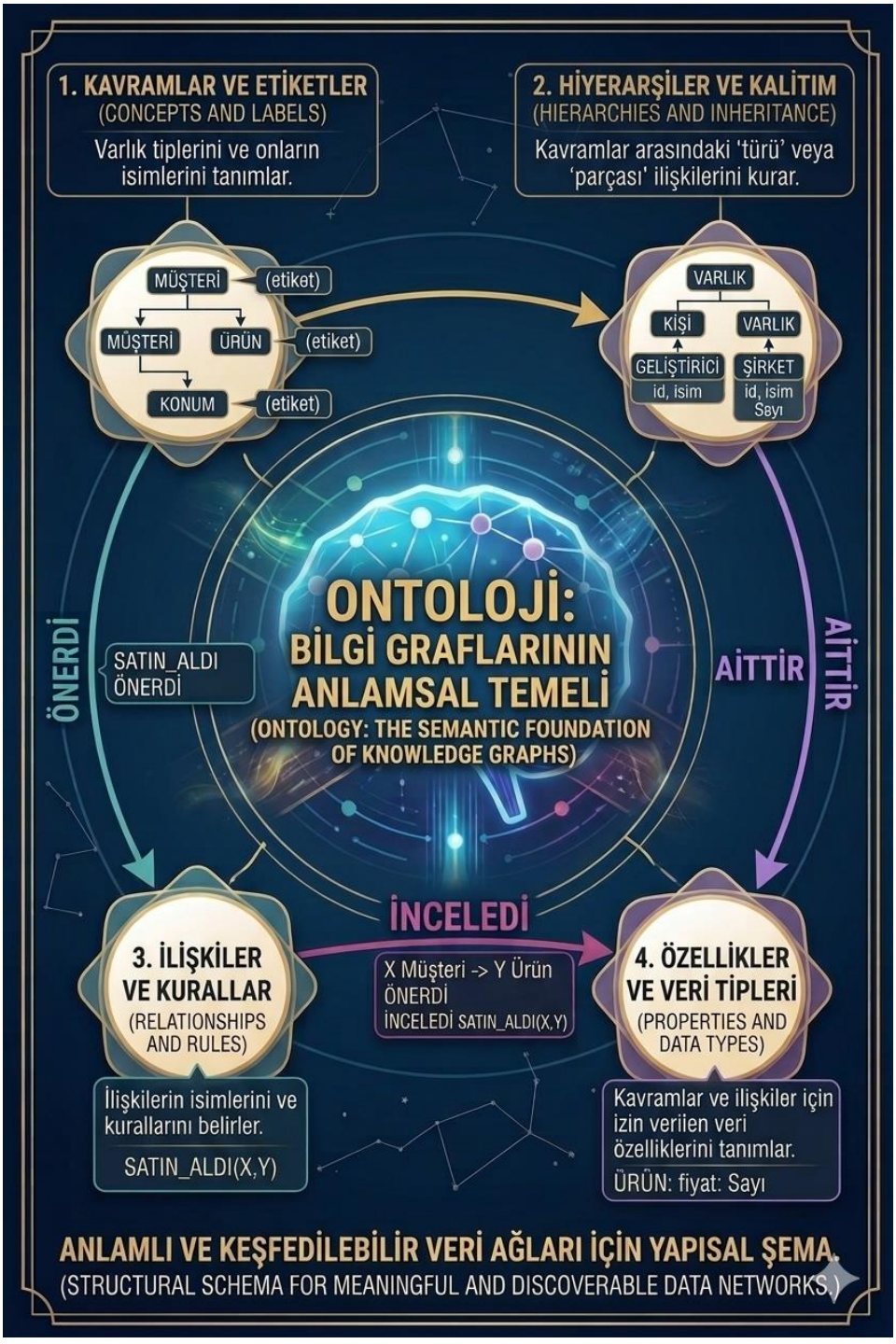
Böylece ontoloji, felsefenin soyut bir tartışma alanından çıkarak dijital çağın temel bilgi mimarilerinden biri haline gelmiştir. Antik filozofların “Varlık nasıl sınıflandırılır?” sorusu bugün yeni bir biçimde karşımıza çıkar: “Bilgiyi makinelerin anlayabileceği şekilde nasıl temsil ederiz?” Modern ontolojiler, işte bu kadim felsefi sorunun teknoloji çağındaki devamıdır.

Ontoloji: Bir Modelleme Dili

Ontoloji, bilgi grafı sistemlerinin en temel yapı taşlarından biridir. Eğer bilgi grafını büyük bir şehir olarak düşünürsek, ontoloji bu şehrin imar planıdır. Hangi yapıların var olacağı, bu yapıların hangi işlevleri üstleneceği ve birbirleriyle nasıl bağlantı kuracağı ontoloji tarafından belirlenir. Bu nedenle ontoloji yalnızca bir veri tanımlama yöntemi değil, aynı zamanda bir modelleme dili olarak görülmelidir. Çünkü ontoloji, gerçek dünyadaki karmaşık ilişkileri sistematik ve makine tarafından anlaşılabilir bir yapıya dönüştürür.

Şekil 6’nın tam merkezinde, veriler arasındaki anlamsal bağların ve kuralların oluşturduğu yapay zekâ zekâsını simgeleyen, neon ışıklarla parıldayan mavi bir insan beyni illüstrasyonu yer alır. Bu merkezi yapının etrafında, ontolojik bir mimarinin olmazsa olmaz bileşenlerini temsil eden dört adet simetrik panel konumlandırılmıştır. Graf sistemlerinde verinin sadece birbiriyle bağlanmasını değil, bu bağlantıların ardındaki "anlam kurallarını" yöneten üst şema (schema) bu paneller aracılığıyla somutlaştırılmıştır.

Şekil 6. Ontoloji



Panellerin detaylı içeriği ve ontolojinin yapısı şu şekildedir:

- **1. Kavramlar ve Etiketler (Concepts and Labels):** Sol üst panelde yer alır ve sistemdeki varlık tiplerini tanımlar. Örnek olarak "Müşteri", "Ürün" ve "Konum" gibi temel sınıfların etiketleme mantığı gösterilmiştir.
- **2. Hiyerarşiler ve Kalıtım (Hierarchies and Inheritance):** Sağ üst panelde, bir önceki görseldeki hatanın düzeltilmesiyle **"Kalıtım"** kavramı netleştirilmiştir. Kavramlar

arasındaki "türüdür" (is-a) veya "parçasıdır" (part-of) ilişkilerini bir soy ağacı gibi kurgular. "Varlık" çatısı altında "Kişi -> Geliştirici" veya "Şirket" gibi alt-üst sınıf hiyerarşilerinin nasıl miras alındığı (inheritance) şematize edilmiştir.

- **3. İlişkiler ve Kurallar (Relationships and Rules):** Sol alt panelde, düğümler arasında kurulabilecek ilişkilerin isimlerini ve mantıksal kurallarını belirler. Örneğin, hangi varlıkların hangileriyle SATIN_ALDI veya ÖNERDİ ilişkisi kurabileceğine dair kural tanımlamalarını içerir.
- **4. Özellikler ve Veri Tipleri (Properties and Data Types):** Sağ alt panelde ise kavramların ve ilişkilerin hangi veri formatlarında nitelikler taşıyabileceğini (Örn: Ürünün fiyatının bir "Sayı/Number" olması gerektiği gibi) kurallara bağlar.

Görselin en altında yer alan "Anlamlı ve keşfedilebilir veri ağları için yapısal şema" ifadesi, ontolojinin veri dünyasındaki görevini özetler: Bilgi graflarına bir "akıl" ve "standart" kazandırarak yapay zekâ ve ajan sistemlerinin veriyi insan gibi anlamlandırmasını sağlamak.

Bilgi grafı sistemlerinde veriler rassal tutulmaz. Her veri öğesi belirli bir kavramsal çerçeve içinde anlam kazanır. Ontoloji tam da bu çerçeveyi sağlar. Bir sistemde hangi tür nesnelerin bulunduğu, bu nesnelerin hangi özelliklere sahip olduğu ve aralarındaki ilişkilerin nasıl kurulacağı önceden tanımlanır. Böylece veri yalnızca depolanan bir unsur olmaktan çıkar; anlamlı bir bilgi ağına dönüşür.

Örneğin akademik yayınları içeren bir bilgi grafı düşünelim. Böyle bir sistemde “Yazar”, “Makale”, “Üniversite” ve “Dergi” temel varlık türleri olabilir. Ontoloji bu kavramları yalnızca isim olarak tanımlamaz; aynı zamanda bunların birbirleriyle nasıl ilişkilendiğini de belirler. Bir “yazar” bir “makale” yazabilir, bir “makale” belirli bir “dergide” yayımlanabilir ve bir “yazar” bir “üniversitede” çalışabilir. Böylece sistem, yalnızca tekil kayıtlar tutmak yerine, akademik dünyanın yapısını temsil eden bir kavramsal ağ oluşturur.

Ontolojinin bir diğer önemli yönü, özellikleri standartlaştırmasıdır. Örneğin “makale” varlığının “başlık”, “yayın yılı” ve “DOI numarası” gibi özellikleri olabilir. Benzer şekilde “yazar” için “isim”, “ORCID”, “uzmanlık alanı” ve “e-posta” gibi alanlar tanımlanabilir. Bu standart yapı sayesinde sistemdeki tüm veriler ortak bir dil kullanılarak organize edilir. Böylece farklı veri kaynaklarından gelen bilgiler aynı çatı altında birleştirilebilir.

Bu durum özellikle büyük ölçekli veri entegrasyonunda kritik öneme sahiptir. Çünkü gerçek dünyada veriler çoğu zaman farklı biçimlerde tutulur. Bir veri tabanında “Prof. Dr. Ahmet Kaya” olarak geçen kişi başka bir sistemde “A. Kaya” biçiminde kayıtlı olabilir. Ontoloji, bu farklı temsil biçimlerini ortak kavramsal yapılar altında birleştirerek veri bütünlüğü sağlar. Böylece sistem, aynı kişiye ait bilgileri ilişkilendirebilir ve daha doğru sonuçlar üretebilir.

Ontoloji aynı zamanda makinelerin akıl yürütmesini kolaylaştırır. Çünkü ontolojide yalnızca nesneler değil, mantıksal kurallar da tanımlanabilir. Örneğin sistemde “Her profesör bir akademisyendir” kuralı varsa, bir kişinin profesör olduğu bilgisi girildiğinde sistem otomatik

olarak onun akademisyen olduğunu da çıkarabilir. Bu özellik, ontolojiyi klasik veri tabanlarından ayıran en önemli unsurlardan biridir.

Modern yapay zekâ sistemlerinde ontolojiler giderek daha merkezi bir rol oynamaktadır. Özellikle semantik web, bilgi grafı ve büyük dil modeli uygulamalarında ontolojiler veriye anlam kazandıran temel mekanizmalardır. Çünkü günümüzde problem yalnızca veri toplamak değildir; bu veriler arasında anlamlı ilişkiler kurabilmektir. Ontoloji, tam da bu nedenle dijital çağın "anlam mühendisliği" araçlarından biri olarak görülmektedir.

Sonuç olarak ontoloji, bilgi grafı sistemlerinin görünmeyen mimarisidir. Verinin nasıl organize edileceğini, hangi ilişkilerin kurulacağını ve sistemin dünyayı nasıl temsil edeceğini belirleyen üst düzey bir modeldir. Bu nedenle ontoloji yalnızca teknik bir veri yapısı değil; aynı zamanda gerçekliği dijital ortamda yeniden kurmaya çalışan kavramsal bir modelleme dilidir.

Ontoloji ile Veri Arasındaki Fark

Bilgi grafı sistemlerinde en sık karıştırılan kavramlardan biri ontoloji ile veri arasındaki farktır. İlk bakışta her ikisi de bilgiyle ilişkili görünse de, aslında farklı düzeylerde işlev görürler. Ontoloji, sistemin kavramsal mimarisini tanımlarken; veri, bu mimarinin içine yerleştirilen somut içerikleri temsil eder. Başka bir ifadeyle ontoloji kuralları koyar, veri ise bu kuralların içinde yer alan gerçek örnekleri oluşturur.

Bu ilişkiyi bir şehir planı benzetmesiyle açıklamak mümkündür. Bir şehir planında yolların nereden geçeceği, parkların nerede bulunacağı ve binaların hangi bölgelerde yükseleceği belirlenir. Ancak bu planın kendisi gerçek şehir değildir. Gerçek şehir; sokaklar, insanlar, araçlar ve binalarla dolu yaşayan yapıdır. Ontoloji de benzer biçimde bilgi sisteminin planını oluşturur. Veri ise bu planın içine yerleşen gerçek içeriktir.

Örneğin akademik bir bilgi grafında ontoloji şu tür kurallar tanımlayabilir: "Bir yazar makale yazar", "Bir makale bir dergide yayımlanır" veya "Bir akademisyen bir üniversitede çalışır." Bunlar genel kavramsal ilişkilerdir. Sistemin nasıl organize edileceğini belirlerler fakat henüz belirli bir kişiden ya da belirli bir yayından söz etmezler.

Veri ise bu soyut yapının içini doldurur. Örneğin Albert Einstein'ın 1905 yılında "Hareketli Cisimlerin Elektrodinamiği Üzerine" adlı makalesini yazdığı bilgisi veridir. Burada ontolojide tanımlanan "yazar → makale yazar" ilişkisi, gerçek bir örnek üzerinden somutlaşmıştır. Aynı şekilde bir üniversiteye bağlı akademisyenler, yayımlanan makaleler veya bilimsel dergiler de ontolojik yapının içine yerleşen veri örnekleridir.

Bu ayrımın anlaşılması, bilgi grafı tasarımında kritik öneme sahiptir. Çünkü ontoloji ile veri birbirine karıştırıldığında sistem esnekliğini kaybeder. İyi tasarlanmış bir ontoloji, yeni verilerin sisteme kolayca eklenebilmesini sağlar. Örneğin sisteme yeni bir araştırmacı ya da yeni bir dergi eklendiğinde, mevcut ontolojik yapı değişmeden kullanılmaya devam eder. Böylece sistem büyüdükçe kaotik hale gelmez.

Ontolojinin bir diğer işlevi de veri tutarlılığını sağlamaktır. Eğer ontoloji “Her makalenin bir yayın yılı vardır” kuralını tanımlıyorsa, sisteme eklenen verilerin de bu yapıya uygun olması beklenir. Böylece eksik, çelişkili veya düzensiz veri girişleri azaltılır. Bu durum özellikle büyük veri sistemlerinde son derece önemlidir.

Sonuç olarak ontoloji ile veri arasındaki ilişki, mimari plan ile gerçek bina arasındaki ilişkiye benzer. Ontoloji sistemi anlamlı hale getiren kuramsal çerçeveyi oluşturur; veri ise bu çerçevenin içini dolduran gerçek dünya örneklerini sağlar. Bilgi grafı teknolojisinin gücü de tam olarak bu iki katmanın birlikte çalışmasından doğar.

Ontolojinin Neden Kritik Olduğu

Modern veri çağında kurumların karşılaştığı en büyük sorunlardan biri artık veri eksikliği değildir. Günümüzde şirketler, üniversiteler, devlet kurumları ve dijital platformlar devasa miktarda veri üretmektedir. Asıl problem, bu verilerin farklı sistemlerde farklı anlam yapılarıyla tutulmasıdır. Başka bir ifadeyle sorun “veri azlığı” değil, “anlam dağınıklığıdır.” İşte ontolojiyi kritik hale getiren temel neden de budur.

Gerçek dünyada aynı kavram çok farklı biçimlerde ifade edilebilir. Örneğin bir bankacılık sisteminde “müşteri” kavramı kullanılırken, başka bir CRM yazılımında aynı varlık “customer”, bir başka sistemde ise “client” olarak tanımlanabilir. İnsan için bu kavramların aynı anlama geldiğini anlamak çoğu zaman kolaydır. Ancak bilgisayar sistemleri açısından bu farklı etiketler, birbirinden tamamen ayrı nesneler gibi görünebilir. Sonuç olarak aynı kişiye ait bilgiler farklı veri kümelerinde parçalanır ve sistemler arasında bütünlük sağlamak zorlaşır.

Bu durum yalnızca isim farklılıklarıyla sınırlı değildir. Tarih formatları, adres yazım biçimleri, kurum isimleri ve ürün tanımları da sistemden sisteme değişebilir. Örneğin bir veri tabanında “İstanbul Teknik Üniversitesi” tam adıyla yer alırken, başka bir sistemde “İTÜ” kısaltması kullanılabilir. İnsan zihni bu bağlantıyı kolayca kurabilir; ancak ontolojik bir yapı olmadan makineler bu iki ifadenin aynı kurumu temsil ettiğini anlayamaz.

Ontoloji tam da bu noktada devreye girer. Ontoloji, farklı veri kaynaklarının ortak bir kavramsal çerçevede buluşmasını sağlar. Yani sistemlere yalnızca veriyi değil, verinin anlamını da öğretir. Böylece “customer”, “client” ve “müşteri” gibi farklı terimlerin aslında aynı kavrama karşılık geldiği tanımlanabilir. Bu yaklaşım, bilgi entegrasyonunun temelini oluşturur.

Özellikle büyük ölçekli organizasyonlarda ontolojinin önemi daha da artar. Çünkü farklı departmanlar çoğu zaman kendi veri standartlarını geliştirir. Pazarlama biriminin kullandığı müşteri tanımı ile finans bölümünün kullandığı tanım birbirinden farklı olabilir. Eğer ortak bir ontoloji yoksa kurum içinde bile veri uyumsuzluğu ortaya çıkar. Bu da yanlış analizlere, eksik raporlara ve hatalı karar süreçlerine yol açabilir.

Ontolojiler yalnızca veri bütünlüğü sağlamakla kalmaz, aynı zamanda yapay zekâ sistemlerinin daha doğru çalışmasına da yardımcı olur. Çünkü yapay zekâ modellerinin kaliteli sonuç üretebilmesi için yalnızca büyük veri değil, anlamlı ve tutarlı veri gerekir. Ontoloji bu anlam

katmanını sağlayarak makinelerin veriler arasındaki ilişkileri daha doğru yorumlamasına olanak verir.

Sonuç olarak ontoloji, modern bilgi sistemlerinin görünmeyen altyapılarından biridir. Veriyi ortak bir anlam evreninde birleştirir, farklı sistemler arasında iletişim kurulmasını sağlar ve bilgi entegrasyonunu mümkün hale getirir. Bu nedenle ontoloji, dijital çağın yalnızca teknik bir aracı değil; aynı zamanda "anlam düzenleme" mekanizmasıdır.

Semantik Katman Olarak Ontoloji

Bilgi sistemlerinde en büyük problemlerden biri, kelimelerin tek bir anlama sahip olmamasıdır. İnsan dili doğası gereği çok katmanlıdır ve aynı sözcük farklı bağlamlarda tamamen farklı şeyleri ifade edebilir. İnsan zihni çoğu zaman bu anlam farklarını sezgisel olarak ayırt eder. Ancak bilgisayar sistemleri için durum çok daha karmaşıktır. İşte ontolojinin "semantik katman" olarak önemi tam bu noktada ortaya çıkar.

Örneğin "Paris" kelimesini ele alalım. Bu ifade çoğu insanın zihninde öncelikle Paris şehrini çağrıştırır. Ancak aynı kelime bir kişi adı olarak da kullanılabilir. Ayrıca "Paris" başlıklı bir film, roman ya da marka adı da olabilir. Eğer bir bilgi sistemi yalnızca kelimeyi görüyorsa, hangi "Paris"ten söz edildiğini anlaması mümkün değildir. Bu durum veri işleme süreçlerinde ciddi belirsizlikler yaratır.

> Ontoloji, yalnızca bir veri modeli değil, aynı zamanda semantik (anlamsal) bir katmandır. Yani sistemin "ne anlama geldiğinin" bilinmesini sağlar.

Ontoloji, bu problemi çözebilmek için kavramlara bağlam kazandırır. Yani yalnızca sözcüğü değil, onun hangi kategoriye ait olduğunu da tanımlar. Örneğin sistemde "Paris" bir şehir olarak tanımlanmışsa, bunun bir coğrafi varlık olduğu belirtilir. Eğer bir kişi adı olarak kullanılıyorsa, "insan" kategorisi altında değerlendirilir. Bir film adı olduğunda ise "sanat eseri" sınıfına bağlanır. Böylece aynı sözcük farklı anlam katmanları içinde ayrıştırılmış olur.

Bu yaklaşım özellikle büyük ölçekli bilgi sistemlerinde kritik önem taşır. Çünkü veri miktarı arttıkça anlam çakışmaları da hızla çoğalır. İnternet, sosyal medya, akademik veri tabanları ve kurumsal arşivler milyarlarca veri üretmektedir. Bu veri evreninde aynı isimlerin, aynı kavramların ve benzer ifadelerin tekrar tekrar ortaya çıkması kaçınılmazdır. Eğer sistem bu kavramları doğru biçimde ayırt edemezse, yanlış eşleştirmeler ve hatalı sonuçlar oluşur.

Örneğin bir arama motoru "Jaguar" kelimesini gördüğünde bunun bir hayvan mı, Jaguar Land Rover markasına ait bir otomobil mi yoksa bir spor takımı mı olduğunu anlamak zorundadır. İnsan bunu bağlamdan çıkarabilir; ancak makineler için bu ayrımı sağlayan yapı ontolojidir. Ontoloji, kavramlar arasındaki ilişkileri tanımlayarak anlam belirsizliğini azaltır.

Semantik katman yaklaşımı, yapay zekâ sistemleri açısından da büyük önem taşır. Çünkü modern yapay zekâ yalnızca veri işlemeye değil, anlam üretmeye çalışmaktadır. Büyük dil

modelleri, bilgi grafı sistemleri ve semantik arama motorları doğru sonuç verebilmek için kavramların bağlamını anlamak zorundadır. Ontoloji, bu sistemlere kavramsal bir harita sunar.

Sonuç olarak ontoloji, veriye yalnızca yapı değil, anlam da kazandırır. Kelimelerin hangi bağlamda kullanıldığını belirleyerek bilgi sistemlerinin daha doğru, daha tutarlı ve daha akıllı çalışmasını sağlar. Bu nedenle ontoloji, modern bilgi çağının görünmeyen semantik altyapılarından biridir.

Ontoloji ve Bilgi Grafı İlişkisi

Ontoloji ile bilgi grafı arasındaki ilişki, modern bilgi sistemlerinin temel mantığını anlamak açısından son derece önemlidir. Bu ilişki çoğu zaman basit ama güçlü bir benzetmeyle açıklanır: Ontoloji iskelettir, bilgi grafı ise bu iskelet üzerine kurulan canlı yapıdır. Başka bir ifadeyle ontoloji sistemin kuramsal çerçevesini oluştururken, bilgi grafı bu çerçeveyi gerçek verilerle doldurur.

Ontoloji, sistemin hangi tür varlıkları kabul edeceğini belirler. Örneğin bir akademik bilgi sisteminde “Yazar”, “Makale”, “Üniversite” ve “Dergi” gibi kavramsal sınıflar ontolojinin parçasıdır. Bunun yanında bu varlıklar arasındaki ilişkiler de ontoloji tarafından tanımlanır. Bir yazarın makale yazabileceği, bir makalenin belirli bir dergide yayımlanabileceği veya bir akademisyenin bir üniversitede çalışabileceği gibi ilişkiler ontolojik yapının içinde yer alır. Böylece sistemin dünyayı nasıl temsil edeceği önceden belirlenmiş olur.

Bilgi grafı ise bu soyut yapının gerçek verilerle doldurulmuş halidir. Örneğin Albert Einstein adlı kişinin bir “Yazar” olduğu, 1905 yılında bir makale yayımladığı ve belirli kurumlarla ilişkilendirildiği bilgileri artık bilgi grafının içeriğini oluşturur. Ontoloji burada kuralları koyarken, bilgi grafı bu kurallar içinde yaşayan veri ağını meydana getirir.

Bu nedenle ontoloji olmadan bilgi grafı çoğu zaman düzensiz bir veri yığınına dönüşebilir. Özellikle büyük ölçekli kurumsal projelerde veri kaynaklarının çokluğu nedeniyle ortak bir kavramsal model zorunlu hale gelir. Çünkü farklı departmanlar veya farklı yazılım sistemleri aynı kavramları farklı biçimlerde tanımlayabilir. Ontoloji, tüm bu verilerin ortak bir anlam dünyasında birleşmesini sağlar.

Graf veri tabanı sistemlerinde ontolojinin rolü kullanılan teknolojiye göre değişebilir. Örneğin Neo4j gibi popüler graf veri tabanı platformlarında ontoloji doğrudan zorunlu değildir. Kullanıcılar düğümleri ve ilişkileri esnek biçimde tanımlayabilirler. Bu durum hızlı prototipleme açısından avantaj sağlasa da, büyük ve karmaşık veri yapılarında standartlaşma sorunları ortaya çıkarabilir. Bu nedenle kurumsal projelerde ontolojik modelleme çoğu zaman kritik hale gelir.

Buna karşılık Semantik Web yaklaşımında ontoloji merkezi bir konuma sahiptir. Özellikle Resource Description Framework (RDF) ve Web Ontology Language (OWL) gibi standartlar, ontolojilerin makineler tarafından anlaşılabilir biçimde tanımlanmasını sağlar. Bu yapı sayesinde sistemler yalnızca veri depolamaz; aynı zamanda veriler arasında mantıksal çıkarımlar da yapabilir.

Sonuç olarak ontoloji ve bilgi grafı birbirini tamamlayan iki temel bileşendir. Ontoloji anlamı ve yapıyı belirlerken, bilgi grafı bu yapıyı gerçek dünya verileriyle canlı hale getirir. Modern bilgi sistemlerinin gücü de tam olarak bu iki katmanın birlikte çalışmasından doğar.

Ontoloji Türleri

Ontolojiler, bilgi sistemlerinde kullanılan kavramsal modellerdir; ancak tüm ontolojiler aynı düzeyde değildir. Bazıları son derece genel kavramları tanımlarken, bazıları belirli bir sektöre ya da doğrudan belirli bir uygulamaya odaklanır. Bu nedenle ontolojiler genellikle üç temel seviyede incelenir: üst ontolojiler, alan ontolojileri ve uygulama ontolojileri. Bu sınıflandırma, bilgi sistemlerinin farklı ölçeklerde nasıl yapılandırıldığını anlamak açısından önemlidir.

En üst düzeyde yer alan yapı "üst ontoloji"dir (upper ontology). Üst ontolojiler, gerçek dünyanın en genel ve en soyut kavramlarını içerir. Bunlar belirli bir sektöre ya da uygulamaya bağlı değildir. "Nesne", "olay", "zaman", "mekân", "süreç" veya "ilişki" gibi kavramlar üst ontolojilerin temel yapı taşlarını oluşturur. Amaç, farklı sistemlerin ortak bir kavramsal temel üzerinde iletişim kurabilmesini sağlamaktır. Örneğin sağlık, finans veya lojistik alanında geliştirilen sistemler farklı veri yapılarına sahip olsa da hepsi "zaman" veya "olay" kavramını kullanabilir. Üst ontolojiler bu ortak zemini oluşturur.

Bir alt seviyede ise alan ontolojileri (domain ontology) bulunur. Bunlar belirli bir uzmanlık alanına özgü kavramları tanımlar. Örneğin sağlık ontolojisinde "hasta", "doktor", "tanı", "ilaç" ve "tedavi" gibi kavramlar yer alır. Finans ontolojilerinde ise "hesap", "ödeme", "kredi" ve "risk" gibi terimler öne çıkar. Akademik ontolojilerde "yazar", "makale", "atıf" ve "üniversite" gibi kavramlar tanımlanır. Alan ontolojileri, belirli sektörlerde veri bütünlüğü ve standartlaşma sağlamak açısından kritik öneme sahiptir. Özellikle farklı kurumların aynı alan içinde veri paylaşımı yapabilmesi için ortak ontolojik çerçeveler gereklidir.

En spesifik düzeyde ise uygulama ontolojileri (application ontology) bulunur. Bunlar belirli bir yazılım ya da bilgi sistemi için özel olarak geliştirilir. Örneğin bir e-ticaret platformunda "ürün", "müşteri", "sipariş", "ödeme", "stok" ve "kargo" gibi kavramlar uygulama ontolojisinin parçası olabilir. Bu ontolojiler doğrudan sistemin işleyişine yönelik tasarlanır ve çoğu zaman kurumun özel ihtiyaçlarına göre şekillenir.

Örneğin bir online alışveriş sisteminde "müşteri sipariş verir", "sipariş ürünü içerir" veya "ürün belirli bir kategoriye aittir" gibi ilişkiler tanımlanabilir. Bu yapı sayesinde sistem yalnızca veri saklamaz; aynı zamanda iş süreçlerini mantıksal olarak organize eder.

Sonuç olarak ontoloji türleri, soyut kavramlardan somut uygulamalara uzanan çok katmanlı bir yapı oluşturur. Üst ontolojiler ortak düşünsel zemini sağlarken, alan ontolojileri belirli uzmanlık alanlarını düzenler, uygulama ontolojileri ise gerçek sistemlerin ihtiyaçlarına göre özelleştirilmiş çözümler sunar. Bu çok katmanlı yaklaşım, modern bilgi sistemlerinin hem esnek hem de anlamlı biçimde çalışmasını mümkün hale getirir.

Ontoloji ve Yapay Zekâ

Yapay zekâ teknolojilerinin hızla geliştiği günümüzde ontoloji kavramı yeniden merkezi bir önem kazanmıştır. Özellikle büyük dil modelleri (LLM'ler), insan benzeri metin üretme konusunda olağanüstü başarılar göstermektedir. OpenAI, Google ve Anthropic gibi şirketlerin geliştirdiği modeller; metin yazabilmekte, çeviri yapabilmekte, kod üretebilmekte ve karmaşık sorulara yanıt verebilmektedir. Ancak bu güçlü dil üretme kapasitesine rağmen, büyük dil modellerinin önemli bir zayıflığı vardır: tutarlı ve kalıcı bir dünya modeli oluşturmakta zorlanmaları.

Dil modelleri esas olarak olasılıksal örüntüler üzerinden çalışır. Milyarlarca metin örneğini analiz ederek hangi kelimelerin birbirini takip etme ihtimalinin yüksek olduğunu öğrenirler. Bu nedenle çoğu zaman etkileyici ve akıcı cevaplar üretirler. Ancak bu süreç, her zaman derin kavramsal tutarlılık anlamına gelmez. Bir model aynı kavram hakkında farklı bağlamlarda çelişkili ifadeler üretebilir ya da mantıksal ilişkileri tam olarak koruyamayabilir. Yapay zekâ araştırmalarında "halüsinasyon" olarak adlandırılan sorunların temel nedenlerinden biri de budur.

İşte ontoloji bu noktada yapay zekâya bir düşünsel çerçeve sunar. Ontoloji yalnızca kelimeleri değil, kavramlar arasındaki ilişkileri de tanımlar. Böylece sistem, "doktor", "hasta" ve "tedavi" kavramlarının nasıl bağlantılı olduğunu anlayabilir. Benzer şekilde bir "şehir", bir "ülke"nin içinde yer alır; bir "üniversite" akademisyenleri barındırır; bir "makale" bir "yazar" tarafından yazılır. Ontolojik yapı, bu ilişkileri sistematik hale getirir.

Bu yaklaşım, yapay zekânın yalnızca dil üretmesini değil, anlam ilişkilerini takip etmesini de sağlar. Başka bir ifadeyle ontoloji, yapay zekâya bir tür kavramsal harita sunar. Böylece model yalnızca cümle kuran bir sistem olmaktan çıkar; belirli ölçüde yapılandırılmış bilgi üzerinde akıl yürütebilen bir yapıya dönüşür.

Bu nedenle modern yapay zekâ mimarilerinde üç önemli bileşen giderek daha fazla birlikte kullanılmaktadır: büyük dil modelleri, bilgi grafı sistemleri ve ontolojiler. Büyük dil modelleri doğal dil üretiminde güçlüdür; bilgi grafıları ilişkisel bilgiyi organize eder; ontolojiler ise bu ilişkilerin anlam çerçevesini belirler. Bu üç yapının birleşimi, daha güvenilir ve daha açıklanabilir yapay zekâ sistemleri geliştirilmesini mümkün kılar.

Özellikle sağlık, hukuk, bilimsel araştırma ve kurumsal bilgi yönetimi gibi alanlarda bu yaklaşım büyük önem taşımaktadır. Çünkü bu alanlarda yalnızca akıcı metin üretmek yeterli değildir; kavramsal doğruluk ve mantıksal tutarlılık da gereklidir.

Sonuç olarak ontoloji, yapay zekâ çağında yeniden kritik bir teknoloji haline gelmiştir. İnsan dilinin karmaşıklığını anlamlandırmak ve makinelerin daha tutarlı bilgi sistemleri oluşturmasını sağlamak için ontolojik yapıların rolü giderek büyümektedir.

Ontoloji Bir Harita Gibidir

Ontolojiyi anlamanın en kolay yollarından biri onu bir haritaya benzetmektir. Nasıl ki bir harita, karmaşık bir şehrin düzenini anlaşılır hale getiriyorsa, ontoloji de karmaşık veri dünyasını

anlamlı bir yapıya dönüştürür. Bu benzetmede şehirler varlıkları, yollar ilişkileri, haritanın kuralları ise ontolojiyi temsil eder. Böylece ontoloji, yalnızca veri öğelerini listeleyen bir yapı olmaktan çıkar; veriler arasındaki bağlantıları düzenleyen bir anlam sistemi haline gelir.

Bir şehir haritasını düşünelim. Harita yalnızca binaları göstermez; aynı zamanda hangi yolların hangi bölgeleri birbirine bağladığını, hangi alanların konut, hangilerinin ticaret bölgesi olduğunu da belirler. İnsanlar bu yapı sayesinde yön bulabilir, rotalarını planlayabilir ve şehir içinde anlamlı hareket edebilirler. Eğer harita olmasaydı, şehir yalnızca birbirinden kopuk sokaklar ve binalar yığınına dönüşürdü.

Bilgi sistemlerinde de durum benzerdir. Modern dünyada veri miktarı olağanüstü boyutlara ulaşmıştır. Şirketler, devlet kurumları, üniversiteler ve dijital platformlar sürekli veri üretmektedir. Ancak veri tek başına anlam üretmez. Eğer bu veriler arasında ilişkiler kurulamazsa, sistem yalnızca devasa bir dijital depo haline gelir. Ontoloji işte bu noktada devreye girerek veriye yön veren görünmeyen haritayı oluşturur.

Örneğin bir akademik bilgi grafında “yazar”, “makale”, “üniversite” ve “dergi” gibi varlıklar şehirdeki yapılar gibidir. Bu varlıklar arasındaki “yazar makale yazar” veya “makale dergide yayımlanır” gibi ilişkiler ise yolları temsil eder. Ontoloji, bu ilişkilerin nasıl kurulacağını belirleyen kurallar bütünüdür. Böylece sistem yalnızca veri depolamaz; aynı zamanda bilgi ağının mantığını da kurar.

Ontolojinin en önemli işlevlerinden biri, büyük veri sistemlerinde yön bulmayı mümkün hale getirmesidir. Nasıl ki haritasız bir şehirde kaybolmak kolaysa, ontolojisiz veri sistemlerinde de anlam kaybı yaşanır. Aynı kavram farklı isimlerle temsil edilebilir, ilişkiler belirsizleşebilir ve veri parçalı hale gelebilir. Ontoloji bu karmaşayı azaltarak sistemin ortak bir anlam çerçevesi içinde çalışmasını sağlar.

Bu nedenle ontoloji, bilgi grafı sistemlerinin görünmeyen omurgası olarak kabul edilir. Kullanıcı çoğu zaman doğrudan ontolojiyi görmez; ancak sistemin tutarlı ve anlamlı çalışmasını sağlayan temel yapı odur. Modern dijital dünyada artık yalnızca veri toplamak yeterli değildir. Asıl değer, bu verilerin hangi kavramsal çerçeve içinde organize edildiğinde ortaya çıkar.

Sonuç olarak ontoloji teknik bir veri modeli olmanın ötesinde, dünyayı dijital ortamda nasıl temsil ettiğimizin temel ifadesidir. İnsan zihni gerçekliği nasıl kategorilere ayırıyorsa, ontoloji de makineler için benzer bir anlam haritası oluşturur. Bu nedenle ontoloji, dijital çağın yalnızca teknik değil, aynı zamanda felsefi altyapılarından biridir.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*.
Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*.
Antoniou, G., & van Harmelen, F. (2004). *A Semantic Web Primer*. MIT Press.

1.5 Semantik Web'in Doğuşu

İnternetin ilk ortaya çıktığı dönemlerde temel fikir oldukça basitti: Bilgiyi paylaşmak. Web sayfaları metin, görsel ve bağlantılardan oluşan dijital belgelerdi. Ancak zamanla bu yapı olağanüstü bir hızla büyüdü ve bugün bildiğimiz devasa bilgi evrenine dönüştü. Bu büyüme, beraberinde önemli bir sorunu da getirdi: Bilgi var, ama anlam makine için erişilebilir değil.

Semantik Web (anlamsal ağ) fikri tam da bu noktada ortaya çıktı. Amaç, interneti yalnızca insanlar için okunabilir bir belge sistemi olmaktan çıkarıp, makinelerin de "anlayabileceği" bir bilgi ağına dönüştürmekti.

Web 1.0'dan Web 2.0'a: Belgeden Etkileşime

İnternetin gelişim süreci yalnızca teknolojik bir dönüşüm değil, aynı zamanda bilginin üretim ve paylaşım biçimlerinin değişim hikâyesidir. Bu dönüşüm genellikle Web 1.0, Web 2.0 ve daha sonra ortaya çıkan semantik web yaklaşımları üzerinden açıklanır. Özellikle Web 1.0'dan Web 2.0'a geçiş, internetin "belge merkezli" bir yapıdan "etkileşim merkezli" bir yapıya dönüşmesini temsil eder.

Web 1.0 olarak adlandırılan ilk internet dönemi, büyük ölçüde statik web sayfalarından oluşuyordu. Kullanıcılar internete giriyor, belirli sitelerdeki içerikleri okuyabiliyor fakat sisteme aktif olarak katkı sağlayamıyordu. Bu yapı bir dijital kütüphaneye benziyordu. Web siteleri çoğunlukla şirketler, üniversiteler veya kurumlar tarafından hazırlanıyor; kullanıcı ise yalnızca "okuyucu" rolünde kalıyordu. Sayfalar nadiren güncelleniyor, etkileşim çok sınırlı düzeyde gerçekleşiyordu. İnternet bu dönemde bilgiye erişim açısından devrim yaratmış olsa da, hâlâ tek yönlü bir iletişim ortamıydı.

2000'li yıllarla birlikte ortaya çıkan Web 2.0 anlayışı ise internetin doğasını kökten değiştirdi. Kullanıcı artık yalnızca tüketici değil, aynı zamanda içerik üreticisi haline geldi. Bloglar, forumlar, sosyal medya platformları ve video paylaşım siteleri interneti yaşayan bir ekosisteme dönüştürdü. İnsanlar düşüncelerini yazıyor, fotoğraf ve video paylaşıyor, yorum yapıyor ve sürekli içerik üretiyordu. Böylece internet, merkezi kurumların yönettiği bir yapı olmaktan çıkarak kitlesel katılımın gerçekleştiği dinamik bir ortama dönüştü.

Bu dönüşümde Facebook, YouTube, Twitter ve blog platformları önemli rol oynadı. İnsan etkileşimi internetin merkezine yerleşti. Bir anlamda web artık belgelerden değil, insanlar arasındaki bağlantılardan oluşan bir ağ haline geldi.

Ancak bu büyük dönüşüm yeni bir problemi de beraberinde getirdi: veri patlaması ve anlam karmaşası. Çünkü kullanıcıların ürettiği içerik miktarı olağanüstü boyutlara ulaştı. Her gün milyonlarca yorum, paylaşım, video ve mesaj internete eklenmeye başladı. Bu içerikler insanlar için anlamlı olabilse de makineler açısından büyük ölçüde düzensizdi.

Web 2.0 insan iletişimini artırdı, ancak makinelerin bu veriyi anlamasını kolaylaştırmadı. Aynı kavram farklı biçimlerde ifade ediliyor, içerikler bağlamsal olarak karmaşık hale geliyor ve veri

yapıları standartlaşmıyordu. İnsan zihni esnek olduğu için bu karmaşayı yönetebiliyordu; fakat bilgisayar sistemleri için anlam çıkarımı hâlâ büyük bir sorundu.

İşte bu noktada semantik web, bilgi grafı ve ontoloji yaklaşımları önem kazanmaya başladı. Çünkü internet artık yalnızca bilgi depolayan bir ortam değil, aynı zamanda anlamın organize edilmesi gereken devasa bir dijital evrene dönüşmüştü.

Sorun: Web Bir Belge Ağıdır, Ama Anlam Ağı Değildir

Geleneksel web mimarisi temelde "belgeler ve bağlantılar" üzerine kurulmuş bir sistemdir. Bu yapı, internetin erken dönemlerinde büyük bir devrim yaratmış olsa da, bilgi organizasyonu açısından önemli bir sınırlılığa sahiptir. Web sayfaları birbirine hyperlink'ler aracılığıyla bağlanır; ancak bu bağlantılar çoğu zaman yalnızca yapısal bir ilişkiyi temsil eder, anlamsal bir içerik taşımaz.

Başka bir ifadeyle, web üzerindeki bağlantılar "bu sayfa şu sayfaya gider" bilgisini verir, fakat "bu iki kavram arasında ne tür bir anlam ilişkisi vardır?" sorusuna cevap vermez. Bu durum, interneti güçlü bir belge ağı haline getirirken, onu aynı zamanda zayıf bir anlam ağına dönüştürür.

Örneğin bir web sayfası "Apple" kelimesine link verebilir. Ancak bu bağlantının neyi ifade ettiği sistem tarafından açıkça bilinmez. "Apple" burada bir meyve türü olan elmayı mı temsil etmektedir, yoksa Apple Inc. adlı teknoloji şirketini mi? İnsan kullanıcı bağlamdan bu ayrımı yapabilir. Fakat bilgisayar sistemleri için bu iki anlam arasında hiçbir fark yoktur; her ikisi de yalnızca aynı metin dizisi olarak değerlendirilir.

Bu durum web'in en temel sorunlarından birini ortaya çıkarır: anlam eksikliği. Web sayfaları arasında bağlantılar güçlüdür, ancak bu bağlantıların neyi temsil ettiği belirsizdir. Dolayısıyla sistemler veri düzeyinde oldukça zengin olsa da, anlam düzeyinde oldukça zayıftır. Milyarlarca belge birbirine bağlıdır, ancak bu bağlantıların çoğu semantik bir yapı oluşturmaz.

Bu sorunun temel sonucu şudur: makineler için web, bir "bilgi ağı" değil, bir "doküman ağı" olarak kalır. Bilgi ağında önemli olan kavramlar arasındaki anlam ilişkileridir; ancak belge ağında yalnızca sayfalar arasındaki referanslar vardır. Bu nedenle arama motorları ve bilgi sistemleri çoğu zaman kullanıcı niyetini tam olarak anlayamaz ve yalnızca anahtar kelime eşleşmelerine dayanarak sonuç üretir.

İşte Semantik Web yaklaşımı bu temel problemi çözmeyi amaçlar. Semantic Web fikri, bağlantılara yalnızca yapısal değil, aynı zamanda anlamsal bilgi eklemeyi hedefler. Yani bir bağlantı yalnızca "A sayfasından B sayfasına git" demekle kalmaz; "A kavramı B kavramıyla şu tür bir ilişki içindedir" bilgisini de taşır.

Bu yaklaşım sayesinde web, yalnızca belgelerden oluşan bir yapı olmaktan çıkarak kavramlar arasındaki ilişkilerin de tanımlandığı bir bilgi ekosistemine dönüşür. Böylece makineler yalnızca metinleri değil, metinlerin arkasındaki anlamı da yorumlayabilir hale gelir.

Sonuç olarak geleneksel web güçlü bir belge sistemi oluşturmuş olsa da, anlam üretme konusunda sınırlıdır. Semantik Web ise bu yapıyı genişleterek interneti yalnızca bağlantıların değil, anlamlı ilişkilerin de bulunduğu bir bilgi ağına dönüştürmeyi hedefler.

Semantik Web'in Doğuşu

Semantik Web kavramı, internetin kurucusu olarak kabul edilen Tim Berners-Lee tarafından ortaya atılmıştır. Berners-Lee, Web'in ilk versiyonunun büyük bir başarı olmasına rağmen temel bir sınırlılığa sahip olduğunu görmüştü: web yalnızca insanlar için anlaşılabilir bir doküman sistemi olarak çalışıyordu. Onun vizyonu ise internetin yalnızca insanlar tarafından değil, makineler tarafından da "anlaşılabilen" bir bilgi ağına dönüşmesiydi.

Bu vizyonun temel sorusu oldukça basitti ama etkisi son derece derindi: "Bilgisayarlar web içeriğini yalnızca göstermek yerine, onu anlayabilir mi?" Bu soru, internetin geleceğini yeniden şekillendiren bir düşünsel kırılma noktası haline geldi. Çünkü mevcut web yapısında bilgisayarlar yalnızca metinleri, bağlantıları ve dosyaları işler; ancak bu içeriklerin ne anlama geldiğini kavrayamaz.

Semantik Web fikri, bu sınırlılığı aşmak için yeni bir yaklaşım önerdi. Buna göre veriler yalnızca belge biçiminde saklanmamalı, aynı zamanda anlamlı varlıklar (entities) ve bu varlıklar arasındaki ilişkiler şeklinde modellenmeliydi. Örneğin bir metin içinde "Paris" kelimesi geçtiğinde, sistem bunun bir şehir mi, bir kişi mi yoksa bir marka mı olduğunu açıkça bilebilmeliydi. Benzer şekilde "yazar", "makale" ve "yayın" gibi kavramlar arasındaki ilişkiler de makine tarafından yorumlanabilir hale gelmeliydi.

Bu yaklaşım, verinin yalnızca gösterilmesini değil, aynı zamanda bağlamsal olarak anlaşılmasını hedefler. Semantik Web'de bilgi, birbirinden kopuk belgeler yerine, birbirine bağlı anlam ağları şeklinde organize edilir. Böylece bilgisayarlar yalnızca "ne yazıldığını" değil, "ne kastedildiğini" de çözümleyebilir hale gelir.

Bu vizyonun hayata geçirilmesi için çeşitli teknolojiler geliştirilmiştir. Özellikle RDF (Resource Description Framework), OWL (Web Ontology Language) ve SPARQL gibi standartlar, verinin anlamlı bir yapı içinde temsil edilmesini sağlar. Bu teknolojiler sayesinde veriler yalnızca depolanmaz, aynı zamanda ilişkisel ve mantıksal bir çerçeve içinde işlenebilir.

Semantik Web'in temel amacı, interneti devasa bir belge koleksiyonundan çıkarıp anlamlı bir bilgi ağına dönüştürmektir. Bu dönüşümde ontolojiler kritik bir rol oynar. Çünkü ontolojiler, kavramların nasıl tanımlanacağını ve birbirleriyle nasıl ilişkilendirileceğini belirler. Böylece makineler, veriler arasında yalnızca yüzeysel bağlantılar kurmakla kalmaz, aynı zamanda anlamlı çıkarımlar da yapabilir.

Sonuç olarak Semantik Web, internetin evriminde ikinci büyük sıçrama olarak görülmektedir. Web 1.0 ve Web 2.0'ın ardından gelen bu yaklaşım, dijital dünyanın yalnızca insanlar için değil, makineler için de anlaşılabilir hale gelmesini amaçlar. Bu vizyon, günümüz yapay zekâ ve bilgi grafı sistemlerinin temel felsefi altyapısını oluşturmaya devam etmektedir.

Temel Fikir: Veri + Anlam + Bağlantı

Semantik Web'in en temel düşüncesi, dijital dünyadaki bilgiyi yalnızca depolanmış veriler olarak değil, aynı zamanda anlam ve ilişkiler ağı olarak ele almaktır. Bu yaklaşım, internetin klasik "belge merkezli" yapısından "anlam merkezli" bir yapıya geçişini ifade eder. Bu dönüşümün özünü ise üç temel katman oluşturur: veri, anlam ve bağlantı.

İlk katman olan veri (data), sistemin en ham halidir. Veri; metinler, sayılar, görseller veya kayıtlar şeklinde bulunabilir. Tek başına veri, anlam taşımaz; yalnızca bir gözlem ya da kayıt niteliğindedir. Örneğin "Einstein", "1905" ve "makale" kelimeleri tek başına yalnızca sembollerden ibarettir. Bu seviyede bilgisayarlar için herhangi bir bağlam ya da yorum yoktur.

İkinci katman olan anlam (meaning), verinin ne ifade ettiğini tanımlar. Bu katman, veriye kavramsal bir içerik kazandırır. Örneğin "Einstein" bir kişi, "1905" bir zaman bilgisi ve "makale" bir akademik üretimdir. Bu noktada veri artık yalnızca bir semboller dizisi olmaktan çıkar ve belirli bir gerçek dünya karşılığına bağlanır. Anlam katmanı, bilgiyi yorumlanabilir hale getirir ve farklı verilerin aynı kavramsal çerçevede değerlendirilmesini sağlar.

Üçüncü katman ise bağlantıdır (link). Bu katman, anlamlandırılmış veriler arasındaki ilişkileri tanımlar. Yani yalnızca "ne olduğu" değil, "neyle ilişkili olduğu" da ifade edilir. Örneğin Albert Einstein'ın 1905 yılında bir makale yazması, bir olaylar ve ilişkiler ağı oluşturur. Burada "kişi", "zaman" ve "yayın" kavramları birbirine bağlanır. Bu bağlantılar olmadan veri parçalı kalır; ancak bağlantılar sayesinde bütünsel bir bilgi yapısı ortaya çıkar.

Bu üç katman birlikte çalıştığında, bilgi yalnızca saklanan bir içerik olmaktan çıkar ve anlamlı bir sistem haline gelir. Semantik Web'in gücü de tam olarak bu noktadan gelir: veriyi, anlamı ve ilişkileri aynı çatı altında birleştirmek.

Bu yaklaşım, modern bilgi grafı sistemlerinin de temelini oluşturur. Google ve Microsoft gibi büyük teknoloji şirketleri, arama motorları ve yapay zekâ sistemlerinde bu üçlü yapıyı aktif olarak kullanmaktadır. Çünkü yalnızca veri depolamak yeterli değildir; verinin ne anlama geldiğini ve diğer verilerle nasıl ilişkilendiğini bilmek gerekir.

Sonuç olarak "veri + anlam + bağlantı" modeli, dijital dünyanın temel mimarisini yeniden tanımlayan bir yaklaşımdır. Bu model sayesinde bilgi sistemleri yalnızca veri yığınları olmaktan çıkar, anlamlı ve ilişkisel bir bilgi ekosistemine dönüşür.

RDF: Üçlü Yapı ile Anlam Modelleme

Semantik Web mimarisinin en temel teknik bileşenlerinden biri RDF (Resource Description Framework) modelidir. RDF, veriyi yalnızca metin veya belge olarak değil, anlamlı ve yapılandırılmış ilişkiler ağı olarak temsil etmeyi amaçlar. Bu yaklaşımın temelinde oldukça basit ama güçlü bir fikir vardır: dünya hakkındaki her bilgi, üçlü bir yapı (triplet) şeklinde ifade edilebilir.

RDF, bilgiyi “üçlü yapı (triple)” ile temsil eder:

- Özne (subject)
- Yüklem (predicate)
- Nesne (object)

Bu üçlü yapı genellikle “özne – yüklem – nesne” formatında kurulur. Yani bir varlık, başka bir varlıkla belirli bir ilişki içindedir. Örneğin “Einstein görelilik teorisini geliştirdi” cümlesi, yüzeyde basit bir ifade gibi görünse de RDF açısından çok daha sistematik bir yapıya sahiptir. Bu cümle şu şekilde modellenir: Albert Einstein → geliştirdi → Görelilik Teorisi.

Tablo 1. RDF Üçlü (Triple) Örnekleri Tablosu

Özne (Subject)	**Yüklem (Predicate)**	**Nesne (Object)**	**Açıklama**
Ali	çalışır	Şirket X	Ali’nin iş ilişkisi
İstanbul	başkent değildir	Türkiye	Yanlış/negatif bilgi de ifade edilebilir
Türkiye	başkenttir	Ankara	Coğrafi bilgi
Elma	türüdür	Meyve	Sınıflandırma
Neo4j	kullanır	Graf veri modeli	Teknoloji ilişkisi
Öğrenci	satın alır	Kitap	E-ticaret örneği
Kitap	yazıldı	Yazar A	Yazar-ürün ilişkisi
Google	geliştirir	Arama motoru	Şirket-ürün ilişkisi
Lityum batarya	parçasıdır	Telefon	Ürün bileşeni ilişkisi
Fabrika	üretir	Ürün	Üretim ilişkisi

Bu temsil biçimi, doğal dilin karmaşıklığını sade ama güçlü bir ilişki modeline indirger. Burada “Einstein” özne (subject), “geliştirdi” yüklem (predicate), “Görelilik Teorisi” ise nesne (object) konumundadır. Bu yapı sayesinde bilgi, makineler tarafından kolayca işlenebilir hale gelir. Çünkü artık sistem yalnızca metni değil, metin içindeki ilişkiyi de açık biçimde görmektedir.

RDF modelinin en önemli avantajlarından biri, tüm bilgiyi standart bir formatta temsil edebilmesidir. Bu sayede farklı veri kaynakları birbiriyle uyumlu hale getirilebilir. Örneğin bir veri setinde “Einstein” bir kişi olarak tanımlanırken, başka bir veri setinde aynı varlık farklı bir biçimde temsil edilse bile RDF sayesinde bu bilgiler aynı kavramsal varlık altında birleştirilebilir. Bu özellik, özellikle büyük ölçekli bilgi entegrasyonu için kritik öneme sahiptir.

RDF yapısında her üçlü, bir bilgi grafının temel bir hücresi gibi çalışır. Birçok üçlü bir araya geldiğinde ise karmaşık ve zengin bir bilgi ağı oluşur. Örneğin Einstein yalnızca “geliştirme” ilişkisiyle değil, aynı zamanda “doğdu”, “çalıştı”, “ödül aldı” gibi birçok farklı ilişkiyle diğer varlıklarla bağlantılı hale gelir. Böylece tek bir birey bile çok boyutlu bir bilgi ağı içinde temsil edilir.

Bu yaklaşımın gücü, her bilginin küçük ama anlamlı parçalara ayrılabilmesinden gelir. RDF, karmaşık dünyayı basit ilişki birimlerine indirgerken aynı zamanda bu birimlerin birleşerek

büyük bir anlam ağı oluşturmasını sağlar. Bu nedenle RDF, yalnızca bir veri formatı değil, aynı zamanda bir düşünme modelidir.

Modern bilgi grafı sistemleri de büyük ölçüde RDF mantığı üzerine inşa edilmiştir. Çünkü bilgi grafi dediğimiz yapı aslında milyonlarca RDF üçlüsünün bir araya gelmiş halidir. Her düğüm bir varlığı, her bağlantı ise bu varlıklar arasındaki ilişkiyi temsil eder.

Sonuç olarak RDF, Semantik Web'in temel yapı taşlarından biridir ve bilgiyi anlamlı ilişkiler ağına dönüştürür. "Özne – yüklem – nesne" üçlüsü sayesinde dünya hakkındaki tüm bilgiler, makineler tarafından işlenebilir ve ilişkilendirilebilir bir forma kavuşur. Bu yapı, modern bilgi sistemlerinin ve yapay zekâ uygulamalarının temelini oluşturan en önemli modellerden biridir.

OWL: Kavramsal Dünyanın Tanımı

Semantik Web'in en güçlü ve en soyut bileşenlerinden biri OWL (Web Ontology Language) dilidir. OWL, yalnızca veriyi temsil etmekle kalmaz; aynı zamanda kavramların nasıl tanımlanacağını, nasıl sınıflandırılacağını ve birbirleriyle hangi mantıksal ilişkiler içinde bulunacağını belirler. Bu yönüyle OWL, Semantik Web'in "kavramsal beyni" olarak kabul edilebilir. Çünkü RDF veriyi ilişkilendirirken, OWL bu ilişkilerin hangi kurallara göre işleyeceğini tanımlar.

> OWL (Web Ontology Language), web üzerinde bilgi alanlarını tanımlamak ve varlıklar arasındaki ilişkileri resmi (formal) bir şekilde modellemek için kullanılan bir ontoloji dilidir. RDF üzerine inşa edilir ve makinelerin veriyi yalnızca saklamasını değil, aynı zamanda anlam çıkarmasını (reasoning / akıl yürütme) da sağlar.

OWL'yi anlamanın en kolay yolu, onu bir "mantıksal kurallar sistemi" gibi düşünmektir. İnsan zihni dünyayı kategoriler üzerinden anlamlandırır: insanlar, meslekler, nesneler, kurumlar ve olaylar gibi. OWL ise bu kategorileri bilgisayarlar için açık ve formal bir yapıya dönüştürür. Böylece sistem yalnızca "ne var?" sorusunu değil, "ne neye dönüşebilir?", "ne neyin alt kümesidir?" gibi daha derin soruları da cevaplayabilir hale gelir.

Örneğin basit bir OWL kuralı şu şekilde olabilir: "Tüm fizikçiler bilim insanıdır." Bu ifade, bir sınıf (class) ilişkisini tanımlar. Yani "fizikçi" sınıfına ait her birey, otomatik olarak "bilim insanı" sınıfına da aittir. Bu tür bir kural sayesinde sistem yeni bir fizikçi bilgisi öğrendiğinde, onun aynı zamanda bir bilim insanı olduğunu da çıkarabilir.

Başka bir örnek: "Her makale bir akademik çalışmadır." Bu kural, "makale" sınıfını daha geniş bir kategori olan "akademik çalışma" sınıfının alt sınıfı olarak tanımlar. Böylece sistem, bir makale gördüğünde bunun akademik üretim alanına ait olduğunu otomatik olarak bilir. Bu tür hiyerarşik ilişkiler OWL'nin en temel kullanım alanlarından biridir.

Benzer şekilde coğrafi bir modelde “bir şehir bir ülkeye aittir” kuralı tanımlanabilir. Örneğin Paris bir şehirdir ve OWL modeli içinde France ülkesine ait olduğu açıkça belirtilir. Bu tür ilişkiler sayesinde sistem, coğrafi verileri yalnızca listelemekle kalmaz, aynı zamanda anlamlı bir hiyerarşi içinde organize eder.

OWL’nin en önemli özelliklerinden biri, yalnızca tanımlayıcı değil aynı zamanda çıkarımsal (inference) bir yapıya sahip olmasıdır. Yani sistem verilen kurallardan yeni bilgiler üretebilir. Örneğin “Tüm profesörler akademisyendir” ve “Ahmet bir profesördür” bilgileri verildiğinde, sistem otomatik olarak “Ahmet bir akademisyendir” sonucuna ulaşabilir. Bu özellik, OWL’yi klasik veri tanımlama dillerinden ayıran en kritik farktır.

OWL, aynı zamanda varlıklar arasındaki karmaşık ilişkileri de tanımlayabilir. Örneğin “bir kişi aynı anda birden fazla üniversitede çalışamaz” gibi kısıtlamalar veya “her makalenin en az bir yazarı olmalıdır” gibi zorunluluklar OWL ile ifade edilebilir. Bu sayede sistem yalnızca doğru veriyi kabul eder, tutarsız bilgileri ise dışlar.

Sonuç olarak OWL, Semantik Web’in kavramsal modelleme dilidir. RDF veriyi ilişkilendirirken, OWL bu ilişkilerin anlamını ve mantıksal çerçevesini oluşturur. Böylece sistemler yalnızca veri depolayan yapılar olmaktan çıkar, dünyayı belirli kurallar ve sınıflar üzerinden modelleyen akıllı bilgi sistemlerine dönüşür.

SPARQL: Anlamlı Sorgulama

Semantik Web’in en önemli bileşenlerinden biri de SPARQL (SPARQL Protocol and RDF Query Language) sistemidir. SPARQL, RDF formatında temsil edilen veriler üzerinde sorgulama yapılmasını sağlayan güçlü bir dildir. Bu yönüyle SPARQL, Semantik Web’in “arama motoru” değil, “anlam motoru” olarak çalışmasını mümkün kılar. Çünkü burada amaç yalnızca kelime eşleştirmek değil, kavramlar arasındaki ilişkileri analiz ederek sonuç üretmektir.

> **SPARQL**, RDF formatında saklanan verileri sorgulamak için kullanılan standart bir sorgu dilidir. Graf veri tabanlarında olduğu gibi, düğüm ve ilişkiler üzerinden “özne–yüklem–nesne” desenleriyle veri üzerinde akıl yürütmeye ve keşif yapmaya imkân verir.

Klasik veri tabanlarında sorgulama genellikle anahtar kelime ya da tablo ilişkileri üzerinden yapılır. Ancak RDF tabanlı yapılarda veri, üçlü yapılar (özne–yüklem–nesne) şeklinde tutulduğu için sorgular da bu ilişki yapısına dayanır. SPARQL, bu üçlüleri hedef alarak çok daha esnek ve anlam odaklı sorgular yapılmasını sağlar. Böylece sistem yalnızca “ne var?” sorusuna değil, “ne ile ne arasında ilişki var?” sorusuna da cevap verebilir.

Örneğin çok basit bir soru düşünelim: “Albert Einstein hangi üniversitelerde çalıştı?” Bu soru SPARQL sayesinde veri grafı üzerinde ilişkisel olarak taranabilir. Sistem, Einstein düğümünden başlayarak “çalıştı” ilişkisi üzerinden bağlı olduğu üniversite düğümlerini bulur ve kullanıcıya sunar. Burada önemli olan nokta, aramanın metin içinde değil, ilişkiler ağı içinde yapılmasıdır.

Benzer şekilde daha karmaşık bir sorgu da oluşturulabilir: “1900 sonrası Nobel alan fizikçiler kimlerdir?” Bu tür bir sorgu, yalnızca bir kelime eşleşmesi değildir. Sistem önce “fizikçi” sınıfını bulur, ardından “Nobel Ödülü” ilişkisini kontrol eder ve son olarak zaman filtresi uygulayarak 1900 sonrası kayıtları seçer. Bu süreç, veri üzerinde çok katmanlı bir mantıksal filtreleme yapılması anlamına gelir.

Bir başka örnek ise kurumsal bilgi sistemlerinde karşımıza çıkar: “Bu şirketle bağlantılı tüm araştırma kurumları nelerdir?” Bu tür bir sorgu, yalnızca isim aramasıyla çözülemez. Sistem, şirket varlığından başlayarak “işbirliği yapar”, “fon sağlar”, “ortak proje yürütür” gibi ilişkileri takip ederek ilgili araştırma kurumlarını tespit eder. Böylece yalnızca doğrudan bağlantılar değil, dolaylı ilişkiler de ortaya çıkar.

SPARQL’in en önemli özelliği, veriyi düz metin olarak değil, bir bilgi grafı üzerinde gezinerek sorgulamasıdır. Bu sayede sistemler yalnızca yüzeysel sonuçlar üretmek yerine, derin ilişkisel analizler yapabilir. Bu yaklaşım, özellikle büyük ölçekli veri kümelerinde anlamlı bilgiye ulaşmayı çok daha güçlü hale getirir.

Örneğin Google gibi büyük teknoloji şirketleri, bilgi grafı tabanlı arama sistemlerinde SPARQL benzeri sorgulama mantıklarını kullanarak kullanıcıların daha doğru ve bağlamsal sonuçlara ulaşmasını sağlar. Çünkü modern bilgi sistemlerinde önemli olan yalnızca veri bulmak değil, doğru anlamı bulabilmektir.

Sonuç olarak SPARQL, Semantik Web’in sorgulama katmanını oluşturan kritik bir teknolojidir. RDF ile modellenmiş veriler üzerinde çalışarak, kullanıcıların yalnızca metin değil, anlam ve ilişki düzeyinde sorular sormasına olanak tanır. Bu nedenle SPARQL, dijital dünyanın klasik arama anlayışını aşarak “ilişkisel bilgi keşfi” çağını başlatan temel araçlardan biridir.

Semantik Web’in Zorlukları

Semantik Web, interneti yalnızca belgelerden oluşan bir yapıdan çıkarıp anlamlı bir bilgi ağına dönüştürmeyi hedefleyen son derece güçlü bir vizyon sunmuştur. Ancak bu vizyon teorik olarak çok çekici olmasına rağmen, pratik uygulamada önemli zorluklarla karşılaşmıştır. Bu zorluklar hem teknik hem de yapısal nitelik taşımaktadır ve Semantik Web’in beklenen hızda yaygınlaşmasını geciktirmiştir.

İlk temel sorun veri üretim maliyetidir. Semantik Web’in çalışabilmesi için verilerin yalnızca yazılması değil, aynı zamanda anlamlı şekilde etiketlenmesi ve yapılandırılması gerekir. Yani her bilginin RDF formatında tanımlanması, ontolojik sınıflara yerleştirilmesi ve ilişkilerinin açıkça belirtilmesi gerekir. Bu süreç, klasik web içerik üretimine kıyasla çok daha zahmetli ve maliyetlidir. Örneğin bir web sayfası oluşturmak kolaydır; ancak aynı içeriği Semantik Web uyumlu hale getirmek ciddi bir emek gerektirir.

İkinci önemli zorluk ontoloji tasarımının karmaşıklığıdır. Ontolojiler, bir alanın tüm kavramsal yapısını modellemeye çalışır. Ancak gerçek dünya son derece karmaşıktır ve sürekli değişir. Bu nedenle doğru ve esnek bir ontoloji tasarlamak oldukça zordur. Yanlış tanımlanmış bir ontoloji,

tüm bilgi sisteminin tutarsız hale gelmesine neden olabilir. Özellikle farklı alanların bir araya geldiği büyük sistemlerde ontoloji tasarımı daha da karmaşık hale gelir.

Üçüncü sorun ise standartların tam anlamıyla yaygınlaşmamış olmasıdır. Her ne kadar RDF, OWL ve SPARQL gibi teknolojiler geliştirilmiş olsa da, bu standartların web'in tamamına entegre edilmesi mümkün olmamıştır. Birçok web sitesi hâlâ yapılandırılmamış veri üretmektedir ve bu veriler Semantik Web'in ihtiyaç duyduğu formata dönüştürülmemiştir.

Dördüncü büyük zorluk ise web içeriğinin doğasıdır. İnternet üzerindeki içeriklerin büyük bir kısmı yapılandırılmamış metinlerden oluşur. Blog yazıları, haberler, sosyal medya içerikleri ve yorumlar doğal dilde yazılmıştır. Bu tür verilerin otomatik olarak anlamlandırılması oldukça zordur. Her ne kadar doğal dil işleme teknikleri gelişmiş olsa da, insan dilinin belirsizliği Semantik Web'in tam anlamıyla uygulanmasını zorlaştırmaktadır.

Tüm bu zorluklar nedeniyle Semantik Web vizyonu uzun süre beklenen hızda ilerleyememiştir. Bir dönem bu fikir "çok idealist ama uygulanması zor" olarak değerlendirilmiştir. Ancak bu, fikrin ortadan kalktığı anlamına gelmez.

Aksine Semantik Web fikri tamamen yok olmamış, farklı bir formda yeniden ortaya çıkmıştır. Bugün bilgi grafı sistemleri, yapay zekâ modelleri ve büyük veri platformları bu vizyonun parçalarını farklı şekillerde hayata geçirmektedir. Özellikle modern yapay zekâ sistemlerinde ontoloji ve ilişki tabanlı veri modelleri giderek daha fazla önem kazanmaktadır.

Sonuç olarak Semantik Web, pratikte karşılaştığı zorluklara rağmen dijital dünyanın gelişiminde temel bir düşünsel çerçeve olmaya devam etmektedir.

Semantik Web'den Bilgi Grafına Geçiş

Semantik Web, interneti yalnızca belgelerden oluşan bir yapı olmaktan çıkarıp anlam temelli bir bilgi ekosistemine dönüştürmeyi hedefleyen güçlü bir vizyondu. Ancak bu vizyonun tüm web ölçeğinde uygulanması oldukça zor olduğu için zamanla farklı bir mühendislik yaklaşımı ortaya çıktı: bilgi grafı. Bu geçiş, teorik bir idealden pratik bir veri mimarisine doğru önemli bir evrimi temsil eder.

Semantik Web'in temel amacı, veriyi anlamlı varlıklar, ilişkiler ve ontolojik yapılar üzerinden modellemekti. Ancak bu yaklaşımın tüm internet üzerinde standart hale getirilmesi mümkün olmadı. Bunun yerine büyük teknoloji şirketleri, Semantik Web'in temel fikirlerini daha kontrollü ve ölçeklenebilir sistemlerde uygulamaya başladı. Böylece "her şeyi semantik hale getirme" hedefi yerini "önemli bilgileri graf yapısında modelleme" yaklaşımına bıraktı.

Bu dönüşümün en önemli örneklerinden biri Google tarafından geliştirilen bilgi grafıdır. Bilgi grafı. Örneğin Google, arama sonuçlarını geliştirmek için büyük ölçekli bir bilgi grafı inşa etti. Bu sistem varlıklar (entities) ve ilişkiler (relations) merkezi hale geldi. Google, arama motoru sonuçlarını daha anlamlı hale getirmek için milyarlarca varlığı ve bunlar arasındaki ilişkileri içeren devasa bir bilgi grafı inşa etti. Bu sistem sayesinde kullanıcılar yalnızca kelime eşleşmesi değil, kavramsal olarak doğru ve bağlamsal sonuçlar elde etmeye başladı. Örneğin bir kişi

“Einstein” aradığında yalnızca metin sayfaları değil, onun çalışmaları, ilişkili kişiler ve bilimsel katkıları da yapılandırılmış biçimde sunulmaktadır.

Benzer şekilde Amazon, ürün öneri sistemlerinde bilgi grafı yaklaşımını kullanarak kullanıcı davranışlarını, ürün özelliklerini ve satın alma ilişkilerini modellemiştir. Bu sayede “bu ürünü alanlar şunları da aldı” gibi öneriler yalnızca istatistiksel değil, ilişkisel bir mantık üzerine kurulmuştur. Ürünler, kullanıcılar ve kategoriler birer düğüm (node) olarak temsil edilirken, aralarındaki etkileşimler bağlantılar (edges) üzerinden modellenir.

Microsoft ise özellikle kurumsal bilgi yönetimi alanında bilgi grafı teknolojilerinden faydalanmaktadır. Şirket içi dokümanlar, çalışanlar, projeler ve departmanlar arasındaki ilişkiler graf yapılarıyla modellenerek bilgiye erişim daha hızlı ve anlamlı hale getirilmiştir. Böylece büyük organizasyonlar içinde bilgi yalnızca saklanmakla kalmaz, aynı zamanda bağlamsal olarak keşfedilebilir hale gelir.

Bu gelişmeler, Semantik Web’in tamamen terk edilmesi anlamına gelmez. Aksine, onun temel fikirlerinin daha uygulanabilir bir mühendislik modeline dönüşmesini ifade eder. Bilgi grafı yaklaşımı, Semantik Web’in idealist hedeflerini daha kontrollü, ölçeklenebilir ve performans odaklı bir yapıya uyarlamıştır. RDF, OWL ve SPARQL gibi teknolojilerin bazıları arka planda kullanılmaya devam ederken, sistemlerin ana mimarisi giderek graf tabanlı modellere kaymıştır.

Bilgi grafı yaklaşımının en büyük avantajı, gerçek dünya problemlerine daha hızlı uyarlanabilmesidir. Büyük veri kümeleri üzerinde çalışırken hem esneklik sağlar hem de performans açısından daha verimli çözümler sunar. Bu nedenle günümüzde yapay zekâ, arama motorları ve öneri sistemlerinin büyük bir kısmı graf tabanlı yapılara dayanır.

Sonuç olarak Semantik Web’den bilgi grafına geçiş, teoriden pratiğe uzanan önemli bir dönüşümdür. Bu dönüşüm, dijital dünyanın yalnızca daha bağlantılı değil, aynı zamanda daha anlamlı hale gelmesini sağlayan temel adımlardan biri olmuştur.

Web Artık Bir Belge Sistemi Değil

Bugün geldiğimiz noktada internetin doğası, Web 1.0 dönemindeki “belge ağı” anlayışından oldukça uzaklaşmıştır. Web artık yalnızca sayfaların birbirine linklerle bağlandığı bir doküman sistemi değildir. Aksine, modern internet; varlıkların (entities), bu varlıklar arasındaki ilişkilerin (relations) ve bunların oluşturduğu ağların (networks) üzerine kurulu devasa bir bilgi grafı haline gelmiştir. Bu dönüşüm, dijital dünyanın yalnızca teknik değil, aynı zamanda kavramsal olarak da yeniden yapılandığını gösterir.

Geleneksel web yapısında bilgi, HTML sayfaları ve hyperlink’ler aracılığıyla organize edilirdi. Bir sayfa başka bir sayfaya bağlantı verir, kullanıcı bu bağlantıları takip ederek bilgiye ulaşırdı. Ancak bu yapı, içeriklerin anlamını açık biçimde temsil etmezdi. Bağlantıların neyi ifade ettiği çoğu zaman belirsizdi. Modern bilgi sistemleri ise bu sınırlılığı aşarak web’i bir anlam ağına dönüştürmeye başlamıştır.

Bugün internet üzerinde her şey bir “varlık” olarak modellenebilir. Bir kişi, bir şirket, bir şehir, bir ürün veya bir kavram; hepsi birer düğüm (node) olarak temsil edilir. Örneğin Google, Amazon veya Microsoft gibi büyük şirketler yalnızca web sayfaları olarak değil, aynı zamanda çok sayıda ilişkiye sahip dijital varlıklar olarak modellenir. Bu varlıklar; ürünlerle, kullanıcılarla, hizmetlerle ve diğer kurumlarla bağlantılıdır.

Bu yapının ikinci temel bileşeni ilişkiler (relations) katmanıdır. İlişkiler, varlıklar arasındaki bağlantıların türünü ve anlamını belirler. Örneğin “satın alır”, “çalışır”, “sahiptir”, “bağlıdır” gibi ilişkiler sayesinde veri yalnızca listelenmez; aynı zamanda yorumlanabilir hale gelir. Böylece sistem, yalnızca “ne var?” sorusunu değil, “ne ile ne arasında nasıl bir ilişki var?” sorusunu da cevaplayabilir.

Üçüncü katman olan ağlar (networks) ise tüm bu varlık ve ilişkilerin birleşiminden oluşur. Büyük ölçekli dijital sistemlerde milyonlarca, hatta milyarlarca düğüm ve bağlantı bir araya gelerek devasa bir graf yapısı oluşturur. Bu yapı içinde bilgi artık doğrusal değil, çok boyutlu ve bağlantısal bir formda organize edilir.

Bu dönüşümün en önemli sonucu, arama motorları ve yapay zekâ sistemlerinin çalışma biçiminin değişmesidir. Modern arama motorları artık yalnızca anahtar kelime eşleştirme yapmaz; bunun yerine bilgi grafı yapıları üzerinden kavramsal sorgulama gerçekleştirir. Kullanıcı bir arama yaptığında sistem yalnızca metinleri değil, varlıklar arasındaki ilişkileri de analiz eder.

Benzer şekilde öneri sistemleri de graf tabanlı yapılardan büyük ölçüde faydalanır. Kullanıcı davranışları, ürün ilişkileri ve içerik bağlantıları bir graf modeli içinde değerlendirilir. Bu sayede daha doğru ve bağlamsal öneriler üretilebilir.

Yapay zekâ modelleri de giderek daha fazla bu graf temelli dünya modelini kullanmaktadır. Çünkü gerçek dünya doğrusal değil, ilişkisel bir yapıya sahiptir. İnsanlar nesneleri ve kavramları ilişkiler üzerinden anlar; modern yapay zekâ sistemleri de bu yapıyı taklit etmeye çalışmaktadır.

Sonuç olarak web artık yalnızca belgelerden oluşan bir sistem değildir. O, varlıkların, ilişkilerin ve ağların oluşturduğu büyük bir bilgi grafıdır. Bu dönüşüm, dijital dünyanın yalnızca nasıl çalıştığını değil, aynı zamanda nasıl düşünüldüğünü de kökten değiştirmiştir.

Semantik Web’in Mirası

Semantik Web, tam anlamıyla ve küresel ölçekte başlangıçta öngörüldüğü biçimde hayata geçirilememiş olsa da, dijital dünyanın düşünsel ve teknik evriminde son derece belirleyici bir rol oynamıştır. Bu vizyon, interneti yalnızca belgelerin ve bağlantıların bulunduğu bir yapı olmaktan çıkarıp, anlamın da açık biçimde temsil edildiği bir bilgi ekosistemine dönüştürmeyi hedeflemiştir. Her ne kadar bu hedefin tüm internet ölçeğinde uygulanması zor olsa da, Semantik Web’in ortaya koyduğu fikirler modern bilgi sistemlerinin temel tasarım ilkelerine dönüşmüştür.

Bugün kullandığımız birçok ileri teknoloji sistem, aslında Semantik Web'in evrimleşmiş ve pratikleştirilmiş versiyonlarıdır. Özellikle bilgi grafı yapıları, bu mirasın en doğrudan devamıdır. Bilgi grafı sistemleri, varlıkları (entities) ve bu varlıklar arasındaki ilişkileri (relations) açık biçimde modelleyerek bilgiyi yalnızca depolamaz, aynı zamanda anlamlı bir ağ haline getirir. Bu yaklaşım, Semantik Web'in temel fikri olan "anlamlı veri temsili" anlayışının doğrudan bir uzantısıdır.

Benzer şekilde yapay zekâ destekli arama motorları da bu mirasın önemli bir parçasıdır. Geleneksel anahtar kelime tabanlı arama sistemlerinin ötesine geçilerek, kullanıcı niyeti ve kavramsal ilişkiler dikkate alınmaya başlanmıştır. Örneğin Google tarafından geliştirilen modern arama altyapıları, yalnızca metin eşleşmesi yapmak yerine varlıklar arasındaki ilişkileri analiz ederek sonuç üretir. Bu sayede arama deneyimi, kelime odaklı olmaktan çıkıp anlam odaklı hale gelir.

Aynı durum büyük dil modelleri için de geçerlidir. OpenAI ve benzeri kurumlar tarafından geliştirilen dil modelleri, büyük veri kümeleri üzerinde eğitilerek dil üretme yeteneği kazanmıştır. Ancak bu modellerin daha güvenilir ve tutarlı hale gelmesi için giderek daha fazla bilgi grafı ve ontolojik yapıdan faydalanılmaktadır. Çünkü yalnızca istatistiksel dil üretimi, her zaman doğru kavramsal tutarlılık sağlamaz.

Öneri sistemleri de Semantik Web fikrinin modern uzantılarından biridir. Amazon ve benzeri platformlar, kullanıcı davranışlarını, ürün ilişkilerini ve içerik bağlantılarını graf yapılar üzerinden modelleyerek daha doğru öneriler üretir. Bu sistemlerde de temel mantık aynıdır: varlıklar ve ilişkiler birlikte değerlendirilir.

Semantik Web'in en önemli mirası ise oldukça net bir felsefi ilkeye dayanır: "Veri tek başına yeterli değildir; anlam ancak ilişkiler aracılığıyla ortaya çıkar." Bu ilke, modern dijital çağın temel paradigmasına dönüşmüştür. Günümüzde hiçbir büyük ölçekli bilgi sistemi yalnızca ham veriyle çalışmamaktadır; veriler mutlaka bağlam, ilişki ve yapı içinde değerlendirilir.

Sonuç olarak Semantik Web, tam anlamıyla uygulanmış bir teknoloji olmaktan çok, modern dijital dünyanın düşünsel altyapısını oluşturan bir fikir sistemi haline gelmiştir. Bugünkü bilgi grafı sistemleri, yapay zekâ uygulamaları ve akıllı algoritmalar, bu vizyonun farklı biçimlerde hayat bulmuş versiyonlarıdır. Semantik Web'in gerçek başarısı da tam olarak burada yatmaktadır: dünyayı yalnızca veri olarak değil, anlamlı ilişkiler bütünü olarak görmemizi sağlamasıdır.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*.

Antoniou, G., & van Harmelen, F. (2004). *A Semantic Web Primer*. MIT Press.

1.6 RDF, OWL ve SPARQL'a Genel Bakış

Semantik Web fikri ortaya çıktıktan sonra en önemli sorunlardan biri şuydu: "Anlamlı bilgi makine tarafından nasıl temsil edilecek?" İnsanlar doğal dil ve sezgiyle kavramlar arasında ilişki kurabilirken, bilgisayarların bu ilişkileri tutarlı bir şekilde saklayıp sorgulayabilmesi için standart bir yapıya ihtiyaç vardı. İşte RDF, OWL ve SPARQL tam olarak bu ihtiyacın üç temel bileşeni olarak ortaya çıktı.

Bu üçlü, Semantik Web'in teorik vizyonunu pratik bir bilgi modelleme ve sorgulama sistemine dönüştürür. RDF veriyi temsil eder, OWL bu verinin kavramsal yapısını tanımlar, SPARQL ise bu yapı üzerinde sorgulama yapılmasını sağlar.

RDF: Üçlü Yapı ile Bilgi Temsili

RDF (Resource Description Framework), Semantik Web'in en temel veri modelleme yaklaşımıdır ve bilgiyi "üçlü yapı" (triple) adı verilen basit ama son derece güçlü bir formatla temsil eder. Bu modelin amacı, karmaşık gerçek dünya bilgisini daha küçük, yönetilebilir ve birbirine bağlanabilir parçalara ayırmaktır. RDF yaklaşımı, bilgiyi üç temel bileşen üzerinden tanımlar: özne (subject), yüklem (predicate) ve nesne (object).

Bu üçlü yapı, aslında dünyanın nasıl temsil edileceğine dair çok temel bir düşünme biçimini ifade eder. Özne, hakkında konuşulan varlığı temsil eder. Yüklem, bu varlık ile başka bir varlık arasındaki ilişkiyi belirtir. Nesne ise bu ilişkinin hedefini ya da bağlandığı ikinci varlığı ifade eder. Bu yapı sayesinde her bilgi, küçük bir ilişki cümlesine dönüştürülür.

Örneğin "Albert Einstein görelilik teorisini geliştirdi" cümlesi RDF formatında şu şekilde temsil edilir: Einstein → geliştirdi → Görelilik Teorisi. Burada Einstein özne, "geliştirdi" yüklem (yani ilişki), Görelilik Teorisi ise nesnedir. Bu basit yapı, aslında çok daha karmaşık bilgi ağlarının temel yapı taşıdır.

RDF modelinin en önemli gücü, tüm bilgiyi bu üçlü yapıların birleşimi olarak ifade edebilmesidir. Tek bir bilgi parçası bile bir üçlüdür; ancak binlerce, milyonlarca üçlü bir araya geldiğinde devasa bir bilgi grafı oluşur. Örneğin:

- Ali → çalışıyor → Üniversite
- Üniversite → bulunuyor → Bursa
- Ali → yazdı → Makale

Bu üç basit ifade, birlikte çok daha zengin bir bilgi yapısı oluşturur. Sistem artık yalnızca "Ali kimdir?" sorusunu değil, "Ali nerede çalışır?", "Hangi şehirde bulunur?", "Ne üretmiştir?" gibi ilişkisel soruları da cevaplayabilir hale gelir.

Bu yapı, klasik ilişkisel veri tabanlarından temelde farklıdır. Geleneksel veri tabanlarında bilgiler tablolar içinde satır ve sütunlar halinde saklanır. Ancak RDF yaklaşımında veri, tablo mantığıyla değil, bir ağ (graph) mantığıyla temsil edilir. Bu nedenle her varlık bir düğüm (node), her ilişki ise bu düğümler arasındaki bir bağlantı (edge) olarak düşünülebilir.

RDF'in bu ağ tabanlı yaklaşımı, özellikle büyük ve heterojen veri kaynaklarının birleştirilmesinde büyük avantaj sağlar. Farklı sistemlerden gelen veriler, ortak bir üçlü yapıya dönüştürüldüğünde kolayca entegre edilebilir. Bu da bilgi bütünlüğünü artırır ve veri silolarını azaltır.

Ayrıca RDF modeli, yalnızca veri saklamakla kalmaz, aynı zamanda veri üzerinde mantıksal çıkarımlar yapılmasına da olanak tanır. Üçlülerin birbirine bağlanması sayesinde yeni ilişkiler keşfedilebilir ve mevcut bilgilerden yeni sonuçlar üretilebilir.

Sonuç olarak RDF, Semantik Web'in temel yapı taşıdır ve bilgiyi basit üçlü ilişkiler üzerinden temsil ederek karmaşık veri dünyasını anlaşılır bir ağ yapısına dönüştürür. Bu yaklaşım, modern bilgi grafı sistemlerinin ve yapay zekâ tabanlı veri modellerinin de temelini oluşturur.

OWL: Kavramların Mantıksal Tanımı

OWL (Web Ontology Language), Semantik Web mimarisinde RDF ile temsil edilen verinin üzerine inşa edilen daha üst düzey bir anlam ve mantık katmanıdır. RDF, bilgiyi özne–yüklem–nesne üçlüleriyle ifade ederek "ne var?" sorusuna yanıt verirken, OWL bu bilginin anlamını derinleştirir ve "bu neyi ifade eder?", "hangi sınıfa aittir?" ve "hangi mantıksal kurallara uymalıdır?" gibi daha soyut sorulara cevap üretir. Bu nedenle OWL, yalnızca bir veri tanımlama dili değil, aynı zamanda bir kavramsal modelleme ve mantıksal çıkarım dilidir.

OWL'nin temel amacı, dünyadaki kavramları bir ontoloji çerçevesi içinde tanımlamak ve bu kavramlar arasındaki ilişkileri resmi (formal) bir yapıya kavuşturmaktır. Bu yapı sayesinde sistemler yalnızca veriyi depolamaz, aynı zamanda bu veriden yeni bilgiler üretebilir. OWL, sınıflar (classes), özellikler (properties) ve kısıtlar (restrictions) aracılığıyla bir alanın mantıksal haritasını oluşturur.

Örneğin OWL kullanılarak şu tür kurallar tanımlanabilir: "Her fizikçi bir bilim insanıdır." Bu ifade, bir sınıf hiyerarşisi oluşturur ve "fizikçi" sınıfının "bilim insanı" sınıfının alt kümesi olduğunu belirtir. Benzer şekilde "Her makale en az bir yazara sahiptir" kuralı, bir zorunluluk (constraint) tanımlar. Bu sayede sistem, yazarı olmayan bir makale gördüğünde bunun ontolojiye aykırı olduğunu anlayabilir.

Bir başka örnek olarak "Bir kişi aynı anda hem öğrenci hem öğretim üyesi olabilir" kuralı, sınıfların birbirini dışlamadığını ifade eder. Bu tür kurallar, gerçek dünyadaki esnek yapının dijital ortama taşınmasını sağlar. OWL, bu tür karmaşık ilişkileri formal mantık kurallarıyla ifade edebilir.

OWL'nin en güçlü yönlerinden biri çıkarım (reasoning) yeteneğidir. Yani sistem, açıkça belirtilmemiş bilgileri mevcut kurallardan türetebilir. Örneğin sistemde şu bilgi varsa: "Ahmet

bir fizikçidir.” OWL kuralları sayesinde sistem otomatik olarak “Ahmet bir bilim insanıdır” sonucuna ulaşabilir. Çünkü “fizikçi → bilim insanı” ilişkisi ontoloji içinde tanımlanmıştır.

Bu çıkarım mekanizması, bilgi grafı sistemlerini yalnızca veri saklayan yapılar olmaktan çıkarır ve onları “mantık yürütebilen” sistemlere dönüştürür. OWL sayesinde sistemler, veriler arasındaki gizli ilişkileri keşfedebilir, tutarsızlıkları tespit edebilir ve yeni bilgi üretebilir.

Örneğin bir akademik bilgi grafında “her makale en az bir yazar içerir” kuralı varsa ve bir makalenin yazarı eksikse, sistem bu durumu otomatik olarak bir hata olarak işaretleyebilir. Bu özellik, veri kalitesini artırmada son derece önemlidir.

OWL ayrıca büyük ölçekli bilgi sistemlerinde standartlaşmayı sağlar. Farklı kaynaklardan gelen veriler, ortak ontolojik kurallar sayesinde uyumlu hale getirilebilir. Bu da özellikle kurumsal bilgi yönetimi ve yapay zekâ sistemleri için kritik bir avantajdır.

Sonuç olarak OWL, Semantik Web’in mantıksal omurgasıdır. RDF veriyi temsil ederken, OWL bu veriye anlam, yapı ve mantık kazandırır. Bu sayede bilgi sistemleri yalnızca pasif veri depolama araçları olmaktan çıkar, aktif olarak çıkarım yapabilen akıllı sistemlere dönüşür.

OWL ve Ontoloji İlişkisi

OWL (Web Ontology Language), ontoloji kavramının bilgisayar bilimleri içindeki en doğrudan ve en resmi karşılığıdır. Ontoloji felsefede varlığın kategorilerini, bu kategoriler arasındaki ilişkileri ve dünyayı nasıl sınıflandırdığımızı inceleyen bir alanken, OWL bu soyut fikri alır ve onu makine tarafından işlenebilir, formal bir dile dönüştürür. Yani OWL, “dünyayı nasıl kavramsallaştırıyoruz?” sorusunu insan zihninden alıp, bilgisayar sistemlerinin anlayabileceği bir mantıksal yapıya çevirir.

OWL’nin temel amacı, bir bilgi alanını kavramsal olarak modellemek ve bu model içinde yer alan tüm varlıkların, sınıfların ve ilişkilerin açık bir şekilde tanımlanmasını sağlamaktır. Bu nedenle OWL, yalnızca veri tanımlamaz; aynı zamanda bu verinin nasıl yorumlanacağını da belirler. Başka bir ifadeyle OWL, bir alanın “kavramsal anayasası” gibi çalışır.

Örneğin bir akademik bilgi sistemi düşünelim. Bu sistemde temel kavramlar arasında “Yazar”, “Makale”, “Dergi” ve “Üniversite” yer alır. OWL, bu kavramları yalnızca listelemekle kalmaz, aynı zamanda onların birbirleriyle olan ilişkilerini de tanımlar. Bir “Yazar” bir “Makale” yazar, bir “Makale” bir “Dergi”de yayımlanır, bir “Yazar” bir “Üniversite” ile ilişkilidir gibi kurallar OWL içinde açıkça modellenir. Bu yapı sayesinde sistem, akademik dünyayı soyut bir model olarak temsil edebilir.

Bu noktada OWL ile RDF arasındaki ilişki oldukça kritik hale gelir. RDF, bilgiyi üçlü yapılar halinde (özne–yüklem–nesne) temsil eder. Ancak RDF tek başına bu yapıların ne anlama geldiğini veya hangi kurallara tabi olduğunu belirtmez. İşte OWL tam da bu boşluğu doldurur. RDF “ne var?” sorusunu yanıtlarken, OWL “bu varlık ne tür bir şeydir?” ve “hangi kurallara uymalıdır?” sorularını cevaplar.

Örneğin RDF düzeyinde şu bilgi olabilir: Albert Einstein → yazdı → Görelilik Teorisi. Bu yalnızca bir ilişkidir. Ancak OWL düzeyinde "her fizikçi bir bilim insanıdır" kuralı tanımlanmışsa, sistem bu bilgiden yola çıkarak Einstein'ın aynı zamanda bir bilim insanı olduğunu da çıkarabilir. Böylece OWL, sistemin yalnızca veri saklamasını değil, aynı zamanda mantıksal çıkarım yapmasını sağlar.

OWL'nin sağladığı bu yapı, bilgi grafı sistemlerinin temel iskeletini oluşturur. Bir bilgi grafında her düğüm (node) bir varlığı, her bağlantı (edge) ise bu varlıklar arasındaki ilişkiyi temsil eder. Ancak bu iskeletin nasıl çalışacağını belirleyen şey OWL kurallarıdır. OWL olmadan bilgi grafı yalnızca bağlantılı verilerden oluşan bir ağ olurdu; OWL ile birlikte bu ağ anlamlı, tutarlı ve mantıksal olarak işleyen bir yapıya dönüşür.

Örneğin bir akademik bilgi grafında "her makale en az bir yazara sahiptir" kuralı OWL ile tanımlanabilir. Bu durumda yazarı olmayan bir makale sistem tarafından tutarsız kabul edilir. Benzer şekilde "bir kişi aynı anda hem öğrenci hem öğretim üyesi olabilir" gibi esnek kurallar da OWL içinde ifade edilebilir. Bu sayede gerçek dünyanın karmaşıklığı dijital ortama daha doğru şekilde yansıtılmış olur.

OWL aynı zamanda veri entegrasyonu açısından da büyük önem taşır. Farklı kurumlardan gelen veriler farklı formatlarda olabilir; ancak ortak bir OWL ontolojisi kullanıldığında bu veriler aynı kavramsal çerçevede birleştirilebilir. Bu durum özellikle büyük ölçekli akademik, sağlık ve kurumsal bilgi sistemlerinde kritik bir avantaj sağlar.

Sonuç olarak OWL, ontoloji kavramının teknik ve formal karşılığıdır. RDF veriyi temsil ederken, OWL bu verinin anlamını, sınıflarını ve kurallarını tanımlar. Böylece bilgi grafı sistemleri yalnızca veri depolayan yapılar olmaktan çıkar, dünyayı kavramsal olarak modelleyen ve bu model üzerinde mantıksal çıkarımlar yapabilen akıllı sistemlere dönüşür.

SPARQL: Bilgi Grafı Sorgulama Dili

SPARQL (SPARQL Protocol and RDF Query Language), Semantik Web ekosisteminin en önemli sorgulama dillerinden biridir ve RDF formatında temsil edilen veriler üzerinde esnek, güçlü ve ilişkisel sorgular yapmayı mümkün kılar. Temel olarak SPARQL, ilişkisel veri tabanlarındaki SQL diline benzer bir rol üstlenir; ancak çalışma mantığı açısından ondan köklü biçimde ayrılır. SQL tablo yapıları üzerinde işlem yaparken, SPARQL doğrudan bir bilgi grafı (graph) yapısı üzerinde çalışır. Bu fark, yalnızca teknik bir ayrıntı değil, aynı zamanda bilgiye yaklaşım biçiminde de önemli bir paradigma değişimini ifade eder.

SPARQL'in en güçlü yönü, veriyi satır ve sütunlardan oluşan düz bir yapı olarak değil, birbirine bağlı varlıklar ve ilişkiler ağı olarak görmesidir. RDF üçlüleri (özne–yüklem–nesne) üzerine kurulu bu yapı sayesinde, her sorgu aslında bir graf üzerinde yapılan yol bulma (graph traversal) işlemine dönüşür. Bu durum, özellikle karmaşık ilişkilerin bulunduğu veri kümelerinde büyük avantaj sağlar.

Örneğin şu tür bir soru düşünelim: “Albert Einstein ile aynı üniversitede çalışmış tüm bilim insanları kimlerdir?” Bu soru, yüzeyde basit gibi görünse de aslında oldukça karmaşık bir ilişkisel sorgudur. Geleneksel ilişkisel veri tabanlarında bu sorunun yanıtı, birçok tablo arasında JOIN işlemleri yapılmasını gerektirir. Einstein’ın çalıştığı üniversiteler bulunur, bu üniversitelerde çalışan diğer kişiler listelenir, ardından bu kişilerin bilim insanı olup olmadığı kontrol edilir. Bu süreç hem maliyetli hem de karmaşıktır.

Buna karşılık RDF tabanlı bir sistemde bu sorgu, doğal bir graf gezintisi haline gelir. Einstein düğümünden başlanarak “çalıştı”, “aynı kurumda bulundu” gibi ilişkiler takip edilir ve ilgili bilim insanı düğümlerine ulaşılır. SPARQL burada yalnızca bir sorgu dili değil, aynı zamanda bir graf keşif aracıdır.

SPARQL sorguları yalnızca veri çekmek için değil, aynı zamanda veri içindeki ilişkileri keşfetmek için de kullanılır. Örneğin “hangi araştırmacılar aynı projelerde yer aldı?”, “hangi şirketler aynı teknoloji alanında iş birliği yaptı?” veya “hangi makaleler birbirini atıf yoluyla etkiledi?” gibi sorular SPARQL ile doğrudan cevaplanabilir. Bu özellik, onu klasik sorgulama dillerinden ayıran en önemli farklardan biridir.

SPARQL’in gücü, verinin yapısını önceden bilmeye gerek duymadan ilişkileri keşfetme yeteneğinden gelir. Kullanıcı, yalnızca sorgulamak istediği kavramları tanımlar; sistem ise graf üzerindeki bağlantıları takip ederek sonuçları üretir. Bu durum, özellikle büyük ve heterojen veri kümelerinde son derece değerlidir.

Örneğin Google gibi büyük ölçekli bilgi sistemleri, arama ve bilgi grafı altyapılarında SPARQL benzeri sorgulama mantıklarından faydalanır. Çünkü modern bilgi sistemlerinde önemli olan yalnızca veri bulmak değil, veriler arasındaki ilişkileri anlamaktır.

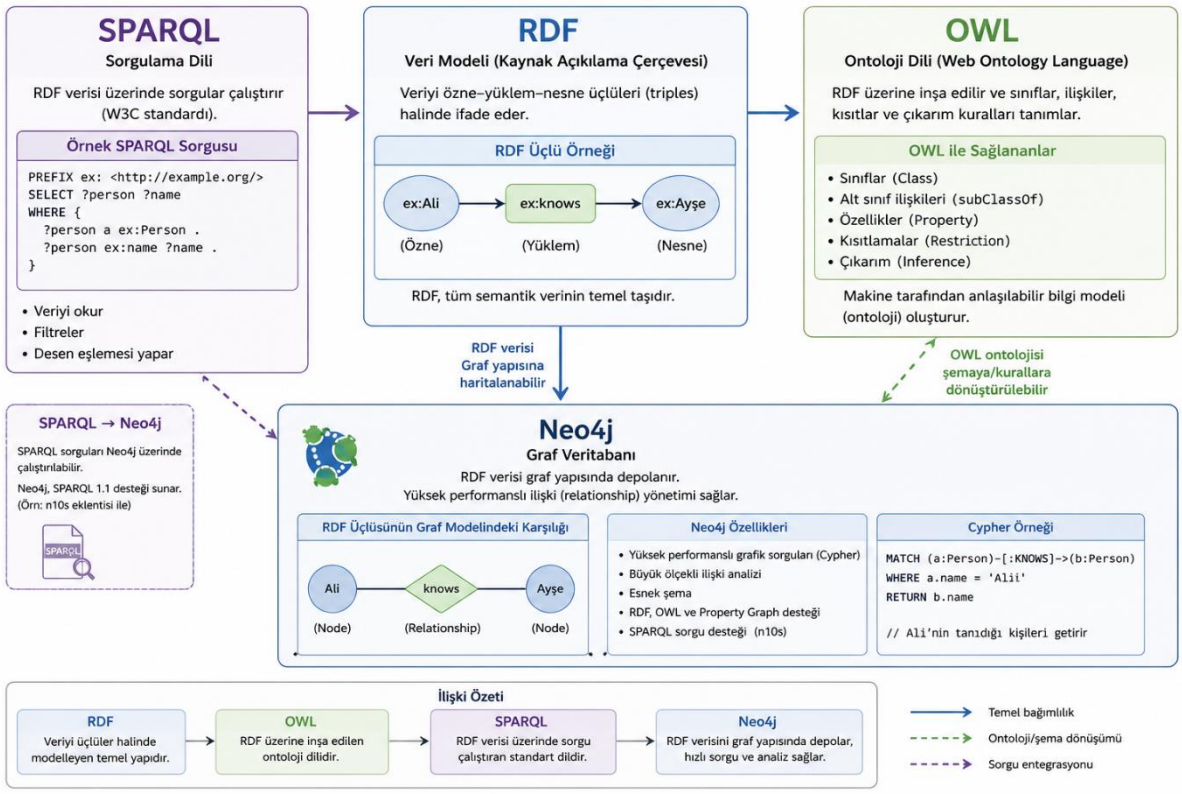
Ayrıca SPARQL, filtreleme, sıralama, gruplama ve mantıksal koşullar gibi gelişmiş sorgulama yeteneklerine de sahiptir. Bu sayede yalnızca basit veri çekme işlemleri değil, aynı zamanda karmaşık analitik sorgular da gerçekleştirilebilir. Örneğin belirli bir zaman aralığında yayınlanmış makaleler, belirli bir kurumla ilişkili araştırmacılar veya belirli bir alan içinde çalışan bilim insanları kolaylıkla bulunabilir.

Sonuç olarak SPARQL, Semantik Web’in sorgulama katmanını oluşturan temel teknolojidir. RDF ile modellenmiş veriler üzerinde çalışarak, klasik veri tabanı mantığını aşar ve bilgiyi bir graf yapısı içinde keşfetme imkânı sunar. Bu yönüyle SPARQL, yalnızca bir sorgu dili değil, aynı zamanda bilgi grafı sistemlerinin analitik düşünme motoru olarak da değerlendirilebilir.

RDF, OWL ve SPARQL Birlikte Nasıl Çalışır?

RDF, OWL ve SPARQL birlikte ele alındığında Semantik Web’in bütünsel mimarisini oluşturan üç katmanlı bir sistem ortaya çıkar. Bu üç bileşen, tek başına güçlü araçlar olsalar da, birlikte kullanıldıklarında veriyi yalnızca depolayan değil, aynı zamanda anlamlandıran ve sorgulayan bir bilgi ekosistemi üretirler. Bu yapı, modern bilgi grafı sistemlerinin de teorik temelini oluşturur.

Şekil 7. RDF, OWL, SPARQL ve Neo4j Arasındaki İlişkiler



Şekil 7, semantik web standartları (**RDF, OWL, SPARQL**) ile modern graf veritabanı mimarisi (**Neo4j**) arasındaki entegrasyonu ve teknik ilişkileri haritalandıran kapsamlı bir akış şemasıdır. Görselin üst şeridinde, W3C standartları olan üç ana bileşen yan yana konumlandırılmıştır: Sol başta RDF verilerini okumak, filtrelemek ve desen eşlemesi yapmak için kullanılan **SPARQL Sorgulama Dili**; ortada veriyi özne-yüklem-nesne üçlüleri (*triples*) halinde ifade eden temel veri modeli **RDF**; sağ başta ise RDF üzerine inşa edilerek sınıflar, hiyerarşiler ve çıkarım kuralları tanımlayan **OWL Ontoloji Dili** yer alır. RDF kutusunun içinde "ex:Ali -> ex:knows -> ex:Ayşe" şeklinde somut bir semantik üçlü örneğine yer verilerek bu teorik standartların alt yapısı görselleştirilmiştir.

Görselin alt yarısı ise bu semantik yapının, yüksek performanslı ilişki yönetimi sağlayan **Neo4j Graf Veritabanı** dünyasındaki karşılığına ayrılmıştır. Şemadaki oklar; RDF verisinin graf yapısına haritalanabildiğini, OWL ontolojilerinin şema ve kurallara dönüştürülebildiğini ve SPARQL sorgularının (özellikle *n10s* eklentisi yardımıyla) Neo4j üzerinde çalıştırılabileceğini net bir şekilde göstermektedir. Neo4j paneli içinde, RDF üçlüsünün özellik grafiğindeki karşılığı (Düğüm-İlişki-Düğüm) gösterilmiş ve aynı mantığın Neo4j'nin kendi sorgu dili olan **Cypher** ile nasıl yazılacağı (MATCH (a:Person)-[:KNOWS]->(b:Person)...) örneklendirilmiştir. En alt bölümde yer alan "İlişki Özeti" akışı ve sağ alttaki gösterge (lejant) kutusu ise tüm bu karmaşık entegrasyonu; temel bağımlılıklar, şema dönüşümleri ve sorgu entegrasyonları olarak üç farklı ok tipiyle özetleyerek mizanpajı tamamlamaktadır.

İlk katman RDF'dir. RDF (Resource Description Framework), verinin temel temsil biçimini sağlar. Bu modelde her bilgi bir "üçlü" (triple) olarak ifade edilir: özne, yüklem ve nesne. Yani dünya hakkındaki her bilgi, basit ilişkisel ifadeler halinde modellenir. Örneğin "Ali çalışıyor Üniversite" ve "Üniversite şehir Bursa" gibi ifadeler RDF düzeyinde temel veri yapılarını oluşturur. RDF, bu sayede bilgiyi graf yapısındaki düğümler ve bağlantılar şeklinde organize eder.

İkinci katman OWL'dir (Web Ontology Language). OWL, RDF ile temsil edilen verinin üzerine anlam ve mantık katmanı ekler. RDF yalnızca "ne var?" sorusuna yanıt verirken, OWL "bu ne anlama gelir?" ve "hangi kurallara tabidir?" sorularını tanımlar. Örneğin OWL içinde "Her çalışan bir kuruma bağlıdır" gibi bir kural tanımlanabilir. Bu tür kurallar sayesinde sistem, eksik veya örtük bilgileri mantıksal çıkarım yoluyla tamamlayabilir. OWL, veriyi yalnızca saklanabilir bir yapı olmaktan çıkarır ve onu yorumlanabilir bir bilgi sistemine dönüştürür.

Üçüncü katman ise SPARQL'dir. SPARQL, RDF formatında saklanan veriler üzerinde sorgulama ve keşif yapmayı sağlar. Kullanıcılar SPARQL sayesinde karmaşık ilişkisel sorular sorabilir. Örneğin "Bursa'da çalışan tüm kişiler kimlerdir?" gibi bir sorgu, sistemin graf yapısı üzerinde gezilerek cevaplanır. Bu işlem klasik veri tabanı sorgularındaki gibi tablo JOIN işlemlerine dayanmaz; bunun yerine doğrudan düğümler arasındaki bağlantılar takip edilir.

Bu üç bileşen birlikte çalıştığında güçlü bir bilgi işleme döngüsü oluşur. RDF veriyi temsil eder, OWL bu veriye anlam ve mantık kazandırır, SPARQL ise bu anlamlı veriyi sorgular ve keşfeder. Bu yapı, Semantik Web'in temel mimarisinin özünü oluşturur.

Örneğin bir akademik sistem düşünelim. RDF katmanında şu bilgiler yer alabilir: Ali → çalışıyor → Üniversite, Üniversite → şehir → Bursa. OWL katmanında "Her çalışan bir kuruma bağlıdır" gibi bir kural tanımlanır. SPARQL katmanında ise "Bursa'da çalışan tüm kişiler kimlerdir?" sorusu sorulur. Sistem bu üç katmanı birlikte kullanarak doğru ve anlamlı sonuç üretir.

Bu bütünleşik yapı sayesinde veri yalnızca saklanmaz, aynı zamanda aktif olarak işlenir ve yorumlanır. Sistem, açıkça belirtilmeyen bilgileri bile çıkarım yoluyla üretebilir. Bu durum, bilgi sistemlerini statik veri depoları olmaktan çıkarıp dinamik bilgi üretim ortamlarına dönüştürür.

Örneğin Google gibi büyük ölçekli sistemler, arama ve bilgi grafı altyapılarında bu üçlü mantığın farklı varyasyonlarını kullanır. RDF benzeri veri modelleri, OWL benzeri ontolojik yapılar ve SPARQL benzeri sorgulama motorları birlikte çalışarak daha akıllı arama deneyimleri sağlar.

Sonuç olarak RDF, OWL ve SPARQL birlikte Semantik Web'in üç temel direğini oluşturur. Bu yapı sayesinde bilgi sistemleri yalnızca veri depolayan araçlar olmaktan çıkar; anlam üreten, çıkarım yapan ve ilişkileri keşfeden akıllı sistemlere dönüşür.

Pratikte Kullanım ve Sınırlamalar

RDF, OWL ve SPARQL birlikte ele alındığında Semantik Web'in teorik olarak son derece güçlü ve tutarlı bir bilgi mimarisi sunduğu görülür. Bu üçlü yapı; veriyi temsil eden RDF, anlam ve kuralları tanımlayan OWL ve sorgulama imkânı sağlayan SPARQL ile birlikte, ideal bir bilgi grafı ekosistemi oluşturur. Ancak bu teorik güce rağmen, pratik uygulamalarda çeşitli zorluklar

ortaya çıkmaktadır. Bu nedenle Semantik Web yaklaşımı her zaman doğrudan ve tam kapsamlı biçimde endüstriyel sistemlere uygulanmamıştır.

İlk önemli zorluk RDF veri modelinin doğasıyla ilgilidir. RDF esnek bir yapıya sahiptir ve her bilgiyi üçlü (subject–predicate–object) formatında temsil eder. Bu yapı küçük ve orta ölçekli veri setlerinde oldukça düzenli ve anlaşılırdır. Ancak veri miktarı arttıkça, milyonlarca hatta milyarlarca üçlünün bir araya gelmesi sistemin karmaşıklığını ciddi şekilde artırır. Özellikle büyük ölçekli bilgi grafı uygulamalarında RDF verisinin yönetimi, performans ve bakım açısından önemli mühendislik sorunları yaratabilir. Bağlantıların artmasıyla birlikte graf yapısının okunabilirliği ve kontrolü zorlaşabilir.

İkinci önemli zorluk OWL tarafında ortaya çıkar. OWL, kavramsal modelleri ve mantıksal kuralları tanımlamak için son derece güçlü bir dil olsa da, doğru bir ontoloji tasarımı oldukça uzmanlık gerektirir. Bir alanın tüm kavramsal yapısını doğru şekilde modellemek; sınıfları, alt sınıfları, ilişkileri ve kısıtlamaları doğru biçimde tanımlamak kolay bir süreç değildir. Yanlış veya eksik tasarlanmış bir ontoloji, tüm sistemde tutarsızlıklara yol açabilir. Bu nedenle OWL tabanlı sistemler genellikle domain uzmanları ve ontoloji mühendisleri tarafından dikkatli bir şekilde tasarlanır.

Üçüncü zorluk SPARQL ile ilgilidir. SPARQL, RDF verisi üzerinde güçlü ve esnek sorgular yapılmasını sağlar; ancak öğrenme eğrisi SQL gibi geleneksel sorgu dillerine kıyasla daha diktir. Kullanıcıların graf yapısını düşünerek sorgu yazması gerekir. Bu da özellikle ilişkisel veri tabanlarına alışkın geliştiriciler için başlangıçta zorlayıcı olabilir. Karmaşık graf sorguları, basit tablo sorgularına göre daha fazla düşünsel modelleme gerektirir.

Bu pratik zorluklar nedeniyle endüstride her zaman tam kapsamlı Semantik Web yaklaşımı kullanılmaz. Özellikle büyük teknoloji şirketleri ve veri yoğun sistemlerde, RDF–OWL–SPARQL üçlüsünün tamamı yerine daha hafif, daha performans odaklı çözümler tercih edilmektedir. Bu noktada bilgi grafı sistemleri öne çıkar.

> Örneğin Neo4j RDF yerine özellik (property) graf modelini kullanarak daha pratik bir yaklaşım sunar. Ancak temel fikir aynıdır: veriyi ilişkiler ağı olarak modellemek.

Örneğin Google ve Amazon gibi şirketler, Semantik Web'in temel fikirlerini benimsemekle birlikte, uygulamada daha ölçeklenebilir ve mühendislik açısından daha yönetilebilir graf veri tabanı yapıları kullanmaktadır. Bu sistemlerde RDF benzeri ilişkisel modeller, OWL benzeri sınıflandırmalar ve SPARQL benzeri sorgulama mantıkları daha hafif formlara indirgenmiştir.

Benzer şekilde Microsoft gibi kurumsal sistemler de bilgi yönetimi çözümlerinde graf tabanlı yapıları tercih ederek performans ve ölçeklenebilirlik arasında denge kurmaktadır. Çünkü gerçek dünya uygulamalarında yalnızca teorik doğruluk değil, aynı zamanda hız, maliyet ve sistem karmaşıklığı da kritik faktörlerdir.

Sonuç olarak RDF, OWL ve SPARQL teorik olarak Semantik Web'in en güçlü bileşenlerini oluşturur. Ancak pratikte karşılaşılan ölçeklenebilirlik, karmaşıklık ve öğrenme zorlukları nedeniyle bu yaklaşım çoğu zaman daha hafif graf tabanlı sistemlere evrilmiştir. Buna rağmen Semantik Web'in temel fikirleri modern bilgi grafı teknolojilerinin çekirdeğinde yaşamaya devam etmektedir.

Sonuç: Anlamın Standartlaştırılması

Modern bilgi teknolojileri incelendiğinde, RDF, OWL ve SPARQL'in doğrudan veya dolaylı etkisinin neredeyse tüm büyük ölçekli bilgi sistemlerine yayıldığı görülür. Her ne kadar Semantik Web'in orijinal vizyonu tam anlamıyla tüm web üzerinde uygulanmamış olsa da, bu üç temel bileşen günümüz bilgi grafı mimarilerinin düşünsel ve teknik altyapısını şekillendirmiştir. Özellikle arama motorları, öneri sistemleri ve yapay zekâ uygulamaları bu yapıların farklı biçimlerde yeniden yorumlanmış versiyonlarını kullanmaktadır.

Modern arama motorları, bilgiyi artık yalnızca metin belgeleri üzerinden değil, "varlıklar" (entities) üzerinden modellemektedir. Bu yaklaşım, RDF'in temel fikrine oldukça yakındır. RDF'de her bilgi bir üçlü yapı (özne–yüklem–nesne) olarak temsil edilirken, arama motorları da benzer şekilde varlıklar arasındaki ilişkileri modelleyerek sonuç üretir. Örneğin bir kullanıcı "Einstein" aradığında sistem yalnızca metin sayfalarını listelemez; onunla ilişkili kişiler, çalışmalar, kurumlar ve kavramları da yapılandırılmış bir biçimde sunar. Bu, bilgiye ilişkisel bir bakış açısının doğrudan sonucudur.

Ontoloji tabanlı sistemler ise OWL'nin temel prensiplerini modern uygulamalara taşımıştır. OWL, kavramların sınıflarını, alt sınıflarını ve aralarındaki mantıksal ilişkileri tanımlayan bir dil olarak tasarlanmıştır. Günümüzde birçok büyük ölçekli veri sisteminde bu mantık, doğrudan OWL kullanılmasa bile benzer şekilde uygulanmaktadır. Örneğin ürün kategorileri, kullanıcı segmentleri veya akademik sınıflandırmalar, hiyerarşik ve mantıksal kurallar çerçevesinde organize edilir. Bu yapı, verinin yalnızca depolanmasını değil, aynı zamanda mantıksal olarak yorumlanmasını da mümkün kılar.

Graf veri tabanları ise SPARQL'in sorgulama mantığını daha düşük seviyeli ve performans odaklı bir şekilde yeniden üretir. SPARQL, RDF üçlüleri üzerinde graf gezintisi (graph traversal) yaparak sorgulama imkânı sunarken, modern graf veri tabanları da benzer şekilde düğümler ve bağlantılar arasında gezerek veri keşfi sağlar. Bu yaklaşım, özellikle karmaşık ilişkilerin bulunduğu büyük veri kümelerinde son derece etkilidir. Kullanıcılar artık yalnızca tablo sorguları değil, ilişki temelli keşifler yapabilmektedir.

Örneğin Google, bilgi grafı tabanlı arama sistemlerinde varlık merkezli bir model kullanarak RDF benzeri temsil yapıları oluşturur. Bu sistemlerde kullanıcı sorguları yalnızca anahtar kelime eşleşmesi üzerinden değil, varlıklar arasındaki ilişkiler üzerinden de değerlendirilir. Benzer şekilde Amazon ve Microsoft gibi şirketler de öneri sistemlerinde graf tabanlı analizlerden yararlanarak kullanıcı davranışlarını ve içerik ilişkilerini modellemektedir.

Yapay zekâ sistemleri ise bu üçlü yapıyı daha soyut ve bütüncül bir biçimde yeniden kullanmaktadır. Büyük dil modelleri, bilgi grafı yapıları ve ontolojik çerçeveler birlikte kullanılarak daha tutarlı, açıklanabilir ve bağlamsal olarak doğru sonuçlar üretilmeye çalışılmaktadır. Bu entegrasyon sayesinde yapay zekâ sistemleri yalnızca metin üreten araçlar olmaktan çıkarak, aynı zamanda ilişkisel bilgi yapıları üzerinde akıl yürütebilen sistemlere dönüşmektedir.

Sonuç olarak RDF, OWL ve SPARQL doğrudan standart olarak kullanılmasa bile, modern bilgi grafı sistemlerinin temel tasarım ilkelerini belirlemiştir. Günümüzde kullanılan arama motorları, veri platformları ve yapay zekâ sistemleri, bu üçlü yapının farklı ölçeklerde yeniden yorumlanmış ve optimize edilmiş versiyonlarıdır. Bu nedenle Semantik Web, pratikte tam olarak gerçekleşmemiş olsa bile, modern dijital dünyanın mimarisini derinden şekillendiren en önemli düşünsel çerçevelerden biri olmaya devam etmektedir.

W3C (2014). *RDF 1.1 Concepts and Abstract Syntax*. https://www.w3.org/TR/rdf11-concepts/

W3C (2012). *OWL 2 Web Ontology Language Document Overview*. https://www.w3.org/TR/owl2-overview/

W3C (2013). *SPARQL 1.1 Query Language*. https://www.w3.org/TR/sparql11-query/

1.7 Bilgi Grafı mı, Graf Veri Tabanı mı?

Graf temelli sistemler üzerine konuşurken en sık karşılaşılan kavramsal hatalardan biri, bilgi grafı (knowledge graph) ile graf veri tabanı (graph database) kavramlarının birbirinin yerine kullanılmasıdır. Oysa bu iki kavram aynı ekosistemin parçaları olsa da farklı soyutlama katmanlarında yer alır. Bu ayrımı netleştirmek, modern veri mimarilerini doğru anlamak için kritik önemdedir.

Bilgi grafı, öncelikle bir modelleme yaklaşımıdır. “Ne temsil ediyoruz?” sorusuna yanıt verir. Burada amaç, dünya hakkındaki varlıkları (örneğin insanlar, kurumlar, ürünler, olaylar) ve bu varlıklar arasındaki ilişkileri anlamlı bir yapıda ifade etmektir. Bu yapı çoğunlukla düğümler (nodes) ve ilişkiler (edges) üzerinden kurulur. Ancak bilgi grafının özü, depolama değil anlam üretimidir. Yani odak noktası verinin semantik olarak organize edilmesi, bağlamın korunması ve çıkarım yapılabilir hale gelmesidir. Ontolojiler, RDF yapıları ve anlamsal kurallar bu katmanın önemli bileşenleridir.

Graf veri tabanı ise daha alt seviyede, mühendislik odaklı bir teknolojidir. “Bunu nasıl saklıyoruz ve nasıl sorguluyoruz?” sorusuna cevap verir. Yani bilgi grafında tanımlanan yapıların fiziksel olarak depolanması, indekslenmesi ve hızlı sorgulanması için kullanılan sistemdir. Neo4j gibi graf veri tabanları, düğümler ve ilişkiler üzerinde optimize edilmiş sorgu dilleri (örneğin Cypher) ile çalışır ve büyük ölçekli ilişkisel veriyi verimli biçimde işleyebilir.

Bu iki kavram arasındaki ilişki hiyerarşiktir: bilgi grafı bir modeldir, graf veri tabanı ise bu modelin uygulanabileceği altyapılardan biridir. Aynı bilgi grafı, farklı veri tabanlarında temsil edilebilir; aynı graf veri tabanı da farklı türden veri modellerini barındırabilir.

Bu ayrımı doğru kurmak, özellikle yapay zekâ, veri bilimi ve kurumsal veri mimarilerinde kritik bir fark yaratır. Çünkü anlam katmanı ile depolama katmanını karıştırmak, sistem tasarımında hem kavramsal hem de teknik hatalara yol açabilir.

Graf Veri Tabanı: Teknik Bir Altyapı

Graf veri tabanı (graf database), veriyi düğümler (node) ve bağlantılar (edge) şeklinde saklayan ve bu yapı üzerinde hızlı sorgulama yapılmasını sağlayan bir veri tabanı türüdür. Buradaki temel amaç, büyük ve karmaşık ilişkisel veriler üzerinde verimli sorgulama yapabilmektir.

Örneğin bir sosyal ağ uygulamasında:

- kullanıcılar düğümdür,
- arkadaşlık ilişkileri bağlantılardır.

Bir kullanıcıdan başlayıp onun arkadaşlarına, onların bağlantılarına hızlıca ulaşmak graf veri tabanlarının doğal gücüdür. Bu sistemler özellikle yoğun ilişkisel sorguların olduğu alanlarda yüksek performans sunmak için tasarlanmıştır.

Neo4j bu yaklaşımın en bilinen örneklerinden biridir. Neo4j gibi sistemler, özellikle kurumsal ölçekli uygulamalarda büyük veri kümeleri üzerinde ilişkisel sorguların hızlı çalışmasını sağlar.

Graf veri tabanlarının temel soruları şunlardır:

- Veriyi nasıl saklarız?
- İlişkileri nasıl hızlı sorgularız?
- Büyük ağlar üzerinde performansı nasıl koruruz?

Bu nedenle graf veri tabanı tamamen mühendislik ve performans odaklı bir katmandır.

Bilgi Grafı: Anlamsal Model

Bilgi grafı (knowledge graph), yalnızca bir veri saklama yöntemi değil, aynı zamanda bir “dünya modeli”dir. Buradaki temel amaç veriyi depolamak değil, gerçek dünyadaki varlıkları ve bu varlıklar arasındaki anlamlı ilişkileri temsil etmektir. Bu yönüyle bilgi grafı, klasik veri tabanı yaklaşımlarından ayrılır; çünkü yalnızca veriyi değil, verinin anlamını da yapılandırır.

Bir bilgi grafında ilk adım, varlıkların (entities) tanımlanmasıdır. Varlıklar, gerçek dünyada karşılığı olan nesneler, kişiler, kurumlar, kavramlar veya olaylar olabilir. Örneğin “Einstein”, “Görelilik Teorisi” ve “Princeton Üniversitesi” birer varlıktır. Ancak bilgi grafının gücü, bu varlıkları tek başına tanımlamakta değil, onları bir bağlam içinde ilişkilendirmektedir.

İkinci adım, bu varlıkların türlerinin (types) belirlenmesidir. “Einstein” bir fizikçidir, yani bir “bilim insanı” türüne aittir. “Görelilik Teorisi” bir bilimsel teoridir, “Princeton Üniversitesi” ise bir eğitim kurumudur. Bu sınıflandırma, veriye yapısal bir düzen kazandırır ve sistemin hangi varlığın ne olduğunu anlamasını sağlar.

Üçüncü ve en kritik adım ise ilişkilerin (relations) anlamlı biçimde modellenmesidir. Bilgi grafında varlıklar yalnızca birbirine bağlanmaz; aynı zamanda bu bağlantının türü ve semantik içeriği açıkça tanımlanır. Örneğin “Einstein → geliştirmiştir → Görelilik Teorisi” ilişkisi, basit bir bağlantı değildir; yönlü, anlamlı ve bağlam taşıyan bir ifadedir. Benzer şekilde “Einstein → çalışmıştır → Princeton Üniversitesi” ilişkisi de sadece bir eşleşme değil, tarihsel ve kurumsal bir bağlam içerir.

Bu yaklaşım sayesinde bilgi grafı, yalnızca “kim kiminle bağlantılı?” sorusuna değil, aynı zamanda “nasıl ve ne tür bir ilişki var?” sorusuna da cevap verebilir. Bu, onu klasik graf yapılardan ayıran temel özelliktir. Çünkü klasik graflarda bağlantılar genellikle yapısaldır; ancak bilgi grafında bağlantılar semantiktir.

Bu nedenle bilgi grafı, bir veri yapısından çok bir semantik modeldir. Yani dünyayı sadece nesneler ve bağlantılar olarak değil, anlam taşıyan ilişkiler ağı olarak temsil eder. Bu yapı, özellikle yapay zekâ, doğal dil işleme ve karar destek sistemleri için büyük bir avantaj sağlar. Çünkü sistem, yalnızca veriyi değil, verinin ne ifade ettiğini de anlayabilir hale gelir.

> Sonuç olarak bilgi grafı, veriyi depolayan bir mekanizma değil; bilgiyi anlamlandıran, bağlamlandıran ve yorumlanabilir hale getiren bir “anlam mimarisi”dir.

Temel Ayrım: Model ve Motor

Bu iki kavram arasındaki en kritik fark şu şekilde özetlenebilir:

- Bilgi grafı = kavramsal model
- Graf veri tabanı (graf database) = çalıştırma motoru

Bir bilgi grafı, kağıt üzerinde bile tasarlanabilir. Örneğin bir şirketin organizasyon yapısı, çalışanları ve projeleri arasındaki ilişkiler çizildiğinde aslında bir bilgi grafı modeli oluşturulmuş olur.

Ancak bu modelin milyonlarca düğüm ve ilişki içeren gerçek sistemlere dönüşmesi için bir graf veri tabanı gerekir.

Neden Karıştırılıyorlar?

Bu iki kavramın sık karıştırılmasının nedeni, pratikte çoğu modern sistemin bu iki katmanı birlikte kullanmasıdır. Gerçek dünya uygulamalarında genellikle:

- önce bilgi grafı tasarlanır (ontoloji ve modelleme),

- sonra bu yapı graf veri tabanına yüklenir,
- ardından sorgular bu altyapı üzerinden çalıştırılır.

Bu nedenle kullanıcı deneyiminde ikisi çoğu zaman tek bir sistem gibi görünür.

Graf Veri Tabanı Bilgi Grafı Olmadan da Vardır

Önemli bir nokta şudur: Bir graf veri tabanı kullanmak, otomatik olarak bir bilgi grafı kurduğunuz anlamına gelmez.

Graf veri tabanları şu tür problemler için de kullanılabilir:

- sosyal ağ ilişkilerini saklamak,
- dolandırıcılık ağlarını analiz etmek,
- öneri sistemleri geliştirmek,
- ağ topolojisi analizleri yapmak.

Bu kullanım senaryolarında sistem yalnızca ilişkileri yönetir, ancak mutlaka bir semantik model içermez. Yani burada "anlam" zorunlu değildir, "bağlantı" yeterlidir.

Bilgi Grafı Graf Veri Tabanı Olmadan da Vardır

Tersi durum da mümkündür. Bir bilgi grafı oluşturmak için mutlaka graf veri tabanı kullanmak gerekmez.

Örneğin:

- RDF (Resource Description Framework) dosyaları,
- JSON-LD yapıları,
- OWL (Web Ontology Language) ontolojileri

ile de bilgi grafı oluşturulabilir.

Ancak bu yaklaşım genellikle daha teorik kalır ve büyük ölçekli sistemlerde performans sınırlamaları ortaya çıkar.

Semantik Katman ve Teknik Katman Ayrımı

Bu ayrımı daha net görmek için iki katmanlı bir yapı düşünmek faydalıdır:

- Semantik katman: Kavramlar, anlamlar, ilişkiler
- Teknik katman: Depolama, sorgulama, performans

Bilgi grafı semantik katmana aittir. Graf veri tabanı ise teknik katmanı temsil eder.

Bu ayrım, modern veri mimarilerinin anlaşılmasında kritik bir zihinsel modeldir.

Kùzu ve Neo4j Yaklaşım Farkı

Küzu ve Neo4j yaklaşım farkı, aslında graf veri tabanı ekosistemindeki iki farklı tasarım felsefesini temsil eder. Bu farkı anlamak, yalnızca hangi aracı kullanacağımızı değil, aynı zamanda veriye nasıl yaklaşacağımızı da belirler.

Neo4j, uzun yıllardır geliştirilmiş, kurumsal dünyada geniş ölçekte kullanılan olgun bir graf veri tabanı sistemidir. Temel tasarım hedefi, büyük hacimli ve sürekli akan ilişkisel veriyi güvenilir, ölçeklenebilir ve yüksek performanslı biçimde yönetmektir. Bu nedenle Neo4j, üretim ortamları için optimize edilmiştir. Finans, telekomünikasyon, lojistik ve sosyal ağ analitiği gibi alanlarda, milyonlarca düğüm ve ilişkiyi gerçek zamanlı sorgulamak gerektiğinde güçlü bir altyapı sunar. Cypher sorgu dili, karmaşık graf yapılarının sezgisel ve okunabilir bir sözdizimiyle sorgulanmasına olanak tanır.

Bildirimsel yaklaşımı sayesinde kullanıcı, sorgunun yürütme ayrıntılarıyla ilgilenmeden yalnızca aradığı ilişki örüntülerini tanımlar. Bu özellik, özellikle bilgi grafı ve ağ analizi uygulamalarında ilişkisel yapıları anlamayı ve analiz etmeyi önemli ölçüde kolaylaştırır. Ayrıca işlem yönetimi, indeksleme, kümeleşme ve yüksek erişilebilirlik gibi özellikler Neo4j'yi kurumsal sistemlerin doğal bir parçası haline getirir.

Buna karşılık yeni nesil bazı graf veri tabanı yaklaşımları, sistemi daha hafif, daha esnek ve veri bilimi odaklı bir yapıya taşımayı hedefler. Bu sistemlerin temel amacı, devasa dağıtık mimariler kurmak değil; daha çok yerel analiz, hızlı deneysel çalışma ve araştırma süreçlerini kolaylaştırmaktır. Bu noktada Kùzu gibi sistemler öne çıkar. Kùzu, genellikle tek makine üzerinde gömülü çalışabilen ve Python ekosistemiyle sıkı entegre bir graf veri tabanı yaklaşımı sunar. Bu özellik, özellikle veri bilimciler ve araştırmacılar için önemli bir avantaj sağlar; çünkü sistem karmaşık kurulumlar gerektirmeden doğrudan analiz süreçlerine entegre edilebilir.

> Kùzu'nun yaklaşımı, "hafif ama güçlü analiz motoru" fikrine dayanır. Büyük ölçekli dağıtık sistemlerden ziyade, hızlı sorgulama, interaktif keşif ve prototipleme süreçlerine odaklanır. Bu nedenle akademik çalışmalar, makine öğrenmesi ön hazırlıkları ve küçük-orta ölçekli bilgi grafı uygulamaları için oldukça uygundur.

Neo4j ve Kùzu arasındaki temel fark, ölçek ve kullanım felsefesinde ortaya çıkar. Neo4j kurumsal süreklilik ve üretim güvenilirliği üzerine inşa edilirken, Kùzu daha çok araştırma esnekliği ve analitik hız üzerine konumlanır. Ancak bu fark, onları birbirinin alternatifi olmaktan çok tamamlayıcı araçlar haline getirir.

Bu ayrım bize kritik bir ilke gösterir: aynı bilgi grafı modeli, farklı graf veri tabanı motorlarında çalıştırılabilir. Yani bilgi grafı kavramsal bir yapı olarak sabittir; değişen şey, bu yapıyı hangi mühendislik ortamında çalıştırdığımızdır. Bir sistem tasarımcısı, üretim ortamı için Neo4j'yi, araştırma ortamı için Kùzu'yu tercih edebilir; ancak temsil edilen bilgi modeli değişmez.

Sonuç olarak bu durum, modern veri mimarilerinde önemli bir esnekliğe işaret eder: model sabit, altyapı değişkendir. Bu esneklik, hem akademik dünyada hem de endüstride bilgi grafı tabanlı sistemlerin hızla yayılmasının temel nedenlerinden biridir.

Hogan, A. et al. (2021). *Knowledge Graphs*. ACM Computing Surveys.

Ehrlinger, L., & Wöß, W. (2016). Towards a Definition of Knowledge Graphs.

Paulheim, H. (2017). Knowledge Graph Refinement: A Survey. *Semantic Web Journal*.

1.8 Neo4j, TigerGraf, Amazon Neptune ve Diğerleri

Graf veri tabanı (graf database) ekosistemi, son on yılda hem akademik araştırmaların hem de endüstriyel ihtiyaçların etkisiyle hızla genişledi. Özellikle büyük veri çağında, ilişkisel yapıların giderek daha önemli hale gelmesi, farklı graf veri tabanı sistemlerinin ortaya çıkmasına yol açtı. Bu sistemlerin her biri aynı temel fikri paylaşır: veriyi düğümler (node) ve bağlantılar (edge) şeklinde temsil etmek. Ancak bu ortak temelin üzerinde, mimari tercihleri, performans hedefleri ve kullanım senaryoları açısından ciddi farklılıklar bulunur. Bu bölümde, modern graf veri tabanı dünyasının en önemli aktörlerine ve bu sistemlerin düşünsel arka planlarına genel bir bakış sunulacaktır.

Neo4j: Graf Veri Tabanlarının Standart Referansı

Bugün dünyadaki en bilinen graf veri tabanı platformlarından biri olan Neo4j, graf veri tabanı dünyasında en yaygın kullanılan ve en olgun sistemlerden biridir. Neo4j geliştirme çalışmaları İsveç'te Emil Eifrem, Johan Svensson ve Peter Neubauer tarafından yaklaşık 2003 yılında başladı. Daha sonra proje olgunlaşarak 2007'de açık kaynaklı bir graf veri tabanı olarak yayımlandı. Neo4j adı da bu dönemde kullanılmaya başlandı. Neo4j, özellik grafı modeli ile öne çıkar. Sistemin temel yeniliği, veriyi tablolar yerine:

- düğümler
- ilişkiler
- özellikler

şeklinde saklaması ve ilişkileri birinci sınıf veri yapısı olarak ele almasıydı. Bu modelde hem düğümler hem de bağlantılar özellik taşıyabilir.

Neo4j'nin en güçlü yönlerinden biri, sorgu dili olarak geliştirdiği Cypher yaklaşımıdır. Cypher, graf yapılar üzerinde doğal dil benzeri bir sorgulama deneyimi sunar. Örneğin bir kullanıcının arkadaşlarının arkadaşlarını bulmak gibi işlemler, sezgisel bir biçimde ifade edilebilir.

Neo4j özellikle şu alanlarda yaygın olarak kullanılır:

- sosyal ağ analizi

- öneri sistemleri
- dolandırıcılık tespiti
- bilgi grafı uygulamaları
- kurumsal veri entegrasyonu

Neo4j'nin yaklaşımı, graf veri tabanını yalnızca bir depolama sistemi değil, aynı zamanda bir analiz ve modelleme aracı haline getirmiştir.

TigerGraf: Büyük Ölçekli Graf Analitiği

TigerGraf, özellikle büyük ölçekli graf analitiği için tasarlanmış bir sistemdir. Neo4j'ye kıyasla daha çok performans ve dağıtık hesaplama odaklıdır. TigerGraf'in temel hedefi, milyarlarca düğüm ve bağlantı içeren graflar üzerinde gerçek zamanlı analitik yapabilmektir. Bu nedenle özellikle finans, telekomünikasyon ve büyük ölçekli sosyal ağ analizlerinde tercih edilir.

TigerGraf'in öne çıkan özellikleri şunlardır:

- paralel graf işleme mimarisi
- gerçek zamanlı analitik yetenek
- yüksek ölçeklenebilirlik
- büyük veri kümeleri üzerinde hızlı gezinme

Bu sistem, graf veri tabanını daha çok "yüksek performanslı analitik motor" olarak konumlandırır.

Amazon Neptune: Bulut Tabanlı Graf Hizmeti

Amazon Neptune, bulut tabanlı bir graf veri tabanı hizmetidir. Amazon Web Services (AWS) ekosistemi içinde yer alır ve kullanıcıların altyapı yönetimiyle uğraşmadan graf veri tabanı kullanmasını sağlar.

Neptune'un en önemli özelliklerinden biri, birden fazla graf modelini desteklemesidir. Hem RDF tabanlı Semantik Web yaklaşımını hem de özellik graf modelini destekleyerek daha esnek bir kullanım sunar.

Neptune özellikle şu senaryolarda tercih edilir:

- bulut tabanlı bilgi grafı uygulamaları
- ölçeklenebilir kurumsal sistemler
- RDF ve SPARQL tabanlı semantik veri yönetimi
- büyük ölçekli veri entegrasyonu

Amazon Neptune'un yaklaşımı, graf veri tabanlarını "servis olarak sunulan altyapı" (Database as a Service) haline getirir.

Kùzu: Gömülü ve Yerel Graf Yaklaşımı

Son yıllarda ortaya çıkan yeni nesil sistemler, graf veri tabanını daha hafif ve geliştirici dostu hale getirmeye odaklanmıştır. Bu yaklaşımın önemli örneklerinden biri Kùzu sistemidir.

Kùzu'nun temel fikri, graf veri tabanını ayrı bir sunucu sistemi olarak değil, doğrudan uygulamanın içine gömülü bir bileşen olarak çalıştırmaktır. Bu yaklaşım, özellikle veri bilimi, akademik araştırmalar ve hızlı prototipleme için büyük avantaj sağlar.

Kùzu'nun öne çıkan yönleri:

- yerel (local-first) çalışma modeli[8]
- Python ile kolay entegrasyon
- düşük kurulum maliyeti
- analitik odaklı tasarım
- hızlı sorgu performansı

Bu yaklaşım, graf veri tabanlarını "hafif veri analitiği araçları" kategorisine yaklaştırır.

RDF Tabanlı Sistemler: Semantik Yaklaşım

Graf veri tabanı dünyasının bir diğer önemli kolu RDF tabanlı sistemlerdir. Bu sistemler özellikle Semantik Web yaklaşımına dayanır ve veriyi üçlü yapı şeklinde temsil eder. Bu kategoride öne çıkan sistemler genellikle SPARQL sorgulama dili ile çalışır ve ontoloji tabanlı veri modellemeye güçlü destek sağlar. RDF tabanlı sistemler daha çok şu alanlarda kullanılır:

- akademik bilgi grafı sistemleri
- biyoinformatik
- kamu veri sistemleri
- semantik arama motorları

Bu yaklaşım, graf veri tabanlarını daha çok "anlamsal veri yönetim sistemi" olarak konumlandırır.

Graf Veri Tabanı Ekosisteminin Çeşitlenmesi

[8] Yerel (local-first) çalışma modeli", bir uygulamanın veriyi ve işleme mantığını mümkün olduğunca kullanıcının kendi cihazında (bilgisayar, telefon vb.) tutarak çalışması demektir. Uygulama internet olmadan da çalışabilir ve veri önce yerelde üretilir, saklanır, sonra gerekiyorsa senkronize edilir.

Bugün graf veri tabanı dünyası tek bir yaklaşım etrafında birleşmiş değildir. Aksine, farklı ihtiyaçlara göre çeşitlenmiş bir ekosistem haline gelmiştir.

Bu ekosistem genel olarak üç ana eksende incelenebilir:

- Kurumsal ve olgun sistemler (Neo4j gibi)
- Yüksek performanslı analitik sistemler (TigerGraf gibi)
- Bulut tabanlı servisler (Amazon Neptune gibi)
- Gömülü ve yerel sistemler (Kùzu gibi)

Bu çeşitlilik, graf veri tabanı yaklaşımının ne kadar geniş bir kullanım alanına sahip olduğunu gösterir.

Sonuç: Aynı Fikir, Farklı Mühendislik Yorumları

Neo4j, TigerGraf, Amazon Neptune ve Kùzu gibi sistemlerin hepsi aynı temel fikre dayanır: veriyi ilişkiler ağı olarak modellemek. Ancak bu fikrin nasıl hayata geçirileceği konusunda farklı mühendislik tercihleri yaparlar.

Bu farkları şu şekilde özetlemek mümkündür:

- Neo4j: olgun, genel amaçlı graf veri tabanı
- TigerGraf: büyük ölçekli graf analitiği
- Amazon Neptune: bulut tabanlı esnek servis
- Kùzu: hafif, gömülü ve yerel analiz sistemi

Bu çeşitlilik bize önemli bir gerçeği gösterir: Graf veri tabanı bir teknoloji ailesidir, tek bir ürün değildir.

Ve bu aile, bilgi grafı çağında giderek daha kritik bir rol oynamaktadır. Çünkü modern dünyada artık yalnızca veri saklamak değil, veriyi ilişkiler üzerinden anlamlandırmak temel ihtiyaç haline gelmiştir.

Neo4j Graph Data Platform Documentation. https://neo4j.com/docs/

Robinson, I., Webber, J., & Eifrem, E. (2015). *Graph Databases*. O'Reilly Media.

Amazon Neptune Documentation. https://docs.aws.amazon.com/neptune/

TigerGraph Documentation. https://docs.tigergraph.com/

1.9 Günlük Hayatta Bilgi grafı Örnekleri

Bilgi grafı kavramı ilk bakışta akademik, teknik ve hatta biraz soyut görünebilir. Ancak aslında günlük hayatımızın içine çoktan yerleşmiş durumdadır. Çoğu insan farkında olmadan her gün onlarca kez bilgi grafı temelli sistemlerle etkileşime girer. Arama motorlarından sosyal medyaya, alışveriş platformlarından dijital asistanlara kadar birçok sistem, dünyayı bir "ilişkiler ağı" olarak modelleyerek çalışır. Bu bölümde bilgi grafı yaklaşımının günlük hayattaki görünmez ama güçlü örneklerini inceleyeceğiz.

Arama Motorları: Bilginin Varlıklar Üzerinden Organize Edilmesi

Bilgi grafının en görünür uygulamalarından biri arama motorlarıdır. Özellikle Google, arama sonuçlarını yalnızca anahtar kelime eşleşmesine göre değil, varlıklar ve ilişkiler üzerinden anlamlandırarak sunar.

Örneğin "Einstein kimdir?" diye arama yaptığınızda sadece metin listesi değil, doğum tarihi, çalıştığı kurumlar, geliştirdiği teoriler gibi yapılandırılmış bilgiler görürsünüz. Bu yapı, klasik metin indekslemenin ötesinde bir bilgi grafı yaklaşımına dayanır.

Google'ın bilgi grafı, şu tür ilişkileri modelleyerek çalışır:

- kişi → doğdu → şehir
- kişi → çalıştı → kurum
- kişi → geliştirdi → teori

Bu sayede arama motoru, yalnızca kelimeleri eşleştirmekle kalmaz; kelimelerin temsil ettiği varlıkları ve bu varlıklar arasındaki ilişkileri de modellemeye çalışır.

Sosyal Medya: İnsanlar ve İlişkiler Grafı

Sosyal medya platformları aslında devasa birer bilgi grafıdır. İnsanlar düğümler, arkadaşlıklar veya takip ilişkileri ise bağlantılar olarak düşünülebilir. Facebook gibi platformlar, kullanıcıları yalnızca bireyler olarak değil, birbirleriyle ilişkili bir ağın parçaları olarak modellemektedir.

Bu yapı sayesinde:

- içerik önerileri,
- arkadaş önerileri,
- reklam hedefleme sistemleri

doğrudan bu ilişkiler ağı üzerinden çalışır.

Bir kullanıcının beğendiği içerik, sadece onun değil, onun bağlantılı olduğu ağın davranışları üzerinden de yorumlanır. Bu da bilgi grafının sosyal davranış analitiğine dönüşmesidir.

Video ve İçerik Platformları: İlgi Grafı

YouTube gibi platformlar, kullanıcıların izleme davranışlarını bir bilgi grafı gibi modelleyerek çalışır. Burada düğümler yalnızca videolar değil, aynı zamanda kullanıcılar, kanallar ve içerik kategorileridir.

İlişkiler ise şu şekilde ortaya çıkar:

- kullanıcı → izledi → video
- video → benzer → video
- kanal → üretti → içerik

Bu yapı sayesinde öneri sistemleri geliştirilir. "Sıradaki video" önerisi aslında bir graf traversal (graf üzerinde gezinme) işlemidir.

E-Ticaret Sistemleri: Ürün İlişkileri Ağı

E-ticaret platformları da bilgi grafı mantığıyla çalışır. Örneğin Amazon, ürünleri yalnızca bağımsız nesneler olarak değil, birbirleriyle ilişkili bir ağ olarak modeller.

Bu ilişkiler şunları içerebilir:

- kullanıcı → satın aldı → ürün
- ürün → birlikte alınır → başka ürün
- ürün → benzer → ürün
- ürün → ait → kategori

Bu yapı sayesinde öneri sistemleri çok daha güçlü hale gelir. "Bu ürünü alanlar bunu da aldı" yaklaşımı aslında graf temelli bir çıkarımdır.

Harita ve Konum Sistemleri: Mekânsal Bilgi Grafı

Harita uygulamaları da bilgi grafının güçlü örneklerinden biridir. Örneğin yollar, şehirler, mekanlar ve kullanıcılar birer düğüm olarak modellenebilir. Apple tarafından geliştirilen harita sistemleri, konumları yalnızca koordinat olarak değil, ilişkisel bir yapı içinde değerlendirir:

- şehir → içerir → mahalle
- yol → bağlar → iki konum
- mekan → bulunur → şehir

Bu sayede en kısa yol bulma, trafik analizi ve konum önerileri gibi işlemler graf algoritmalarıyla yapılır.

Dijital Asistanlar: Kişisel Bilgi Grafı

Siri, Google Assistant ve benzeri dijital asistanlar da bilgi grafı mantığını kullanır. Kullanıcının geçmiş etkileşimleri, tercihleri, takvimi ve alışkanlıkları bir graf yapı içinde tutulur.

Bu yapı sayesinde sistem:

- “Yarın hava nasıl?”
- “Toplantım nerede?”
- “En son konuştuğumuz konu neydi?”

gibi soruları bağlamsal olarak cevaplayabilir.

Burada bilgi grafı, kişisel bir hafıza sistemi gibi çalışır.

Sağlık Sistemleri: Hasta ve Hastalık İlişkileri

Sağlık alanında da bilgi grafı kullanımı giderek artmaktadır. Hastalar, hastalıklar, semptomlar ve tedaviler birer düğüm olarak modellenir.

İlişkiler ise:

- hasta → sahip → hastalık
- hastalık → gösterir → semptom
- ilaç → tedavi eder → hastalık

şeklinde kurulur.

Bu yapı sayesinde teşhis destek sistemleri ve ilaç keşif süreçleri daha sistematik hale gelir.

Günlük Hayatın Görünmez Grafı

Aslında farkında olmadan sürekli bir bilgi grafı içinde yaşıyoruz. Telefonumuzdaki rehber bile bir graf yapısıdır:

- kişi → aradı → kişi
- kişi → mesaj attı → kişi
- kişi → çalışıyor → kurum

Benzer şekilde bankacılık sistemleri, ulaşım ağları ve hatta alışkanlıklarımız bile graf olarak modellenebilir.

Sonuç: Her Şey Bir Ağa Dönüşüyor

Günlük hayattaki tüm bu örnekler bize tek bir şeyi gösterir: modern dijital sistemler dünyayı artık listeler veya tablolar olarak değil, ilişkiler ağı olarak görmektedir.

Bu dönüşümün temel mesajı şudur:

“Veri tek başına anlam taşımaz; anlam, ilişkiler içinde ortaya çıkar.”

Bilgi grafı yaklaşımı, bu ilişkileri görünür hale getirerek dijital dünyanın temel düşünme biçimini değiştirmiştir.

Google AI Blog (2012). Introducing the Knowledge Graph. https://blog.google/products/search/introducing-knowledge-graph/

Singhal, A. (2012). *The Knowledge Graph: Things, Not Strings*. Google Research.

Facebook Engineering Blog (Social Graph architecture). https://engineering.fb.com/

BÖLÜM 2 — GRAFLARLA DÜŞÜNMEK

Bu bölümde amaç, bilgiyi yalnızca satırlar ve sütunlar halinde düzenlenmiş bir veri yapısı olarak değil, birbirine bağlı ilişkiler ağı olarak görme becerisini geliştirmektir. Graflarla düşünmek, dünyayı düğümler ve bağlantılar üzerinden algılamak anlamına gelir. Bu yaklaşım, modern bilgi sistemlerinin temelini oluşturduğu gibi, insan zihninin doğal çalışma biçimine de oldukça yakındır.

Geleneksel veri modelleri çoğunlukla doğrusal veya tablosal yapılara dayanır. Oysa gerçek dünya doğrusal değildir; ilişkiler çok katmanlı, çok yönlü ve dinamik bir ağ yapısı sergiler. Bir kişinin yalnızca kim olduğu değil, kimlerle bağlantılı olduğu, hangi kurumlarda bulunduğu, hangi olaylarla ilişkili olduğu gibi unsurlar birlikte anlam üretir. Graf düşüncesi tam olarak bu çoklu ilişkileri merkeze alır.

Bu bölümde ağ bilimi, küçük dünya grafları (small-world networks), ölçekten bağımsız graflar (scale-free networks) ve merkezilik (centrality) gibi temel kavramlar üzerinden graf düşüncesinin teorik temelleri ele alınacaktır. Ayrıca sosyal ağlardan bilgi graflarına geçiş süreci incelenerek, modern dijital sistemlerin neden graf tabanlı modellere yöneldiği açıklanacaktır.

Graflarla düşünmek yalnızca teknik bir beceri değildir; aynı zamanda bir bilişsel perspektiftir. Bu perspektif, veriyi izole parçalar yerine ilişkiler bütünü olarak görmeyi sağlar. Böylece karmaşık sistemler daha anlaşılır hale gelir ve gizli bağlantılar görünür olur. Bu yaklaşım, özellikle yapay zekâ, bilgi grafı sistemleri ve büyük veri analitiği alanlarında temel bir düşünme biçimi haline gelmiştir.

2.1 İnsan Zihni ve İlişkisel Düşünme

İnsan zihni dünyayı çoğu zaman tekil nesneler üzerinden değil, ilişkiler üzerinden anlamlandırır. Bir kişiyi tanımlarken yalnızca fiziksel özelliklerini değil, onun kimlerle bağlantılı olduğunu, hangi olaylara dahil olduğunu ve hangi bağlamlarda yer aldığını da düşünürüz. Bu durum, zihnin doğal olarak ilişkisel bir düşünme biçimine sahip olduğunu gösterir. Yani bilgi, zihinde izole birimler halinde değil, birbirine bağlanan çağrışım ağları şeklinde örgütlenir.

Bellek yapısı da benzer bir mantıkla çalışır. Bir kavram, başka bir kavramı tetikler; bir anı, başka bir anıyı çağırır. Bu çağrışım zinciri, aslında zihinsel bir graf yapıdır. Örneğin “okul” kavramı, “arkadaşlar”, “öğretmenler”, “sınavlar” ve “gençlik” gibi düğümleri harekete geçirir. Bu yapı, insan zihninin sabit bir veri deposu değil, dinamik bir ilişki ağı olduğunu gösterir.

Modern bilişsel bilimler, insan düşüncesinin bu ağ benzeri yapısını giderek daha fazla vurgulamaktadır. Kavramlar arası bağlantılar güçlendikçe öğrenme kolaylaşır, zayıfladıkça unutma gerçekleşir. Bu nedenle öğrenme süreci, aslında yeni düğümler eklemekten çok mevcut ağın yeniden düzenlenmesi olarak görülebilir.

Bilgi grafı yaklaşımı, insan zihninin bu doğal yapısını taklit etmeye çalışır. Yapay sistemlerde varlıkların ve ilişkilerin modellenmesi, zihinsel süreçlerin dijital bir karşılığı olarak düşünülebilir. Bu nedenle graf tabanlı sistemler, yalnızca teknik bir tercih değil, aynı zamanda bilişsel bir modellemenin sonucudur.

Kahneman, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Tversky, A., & Kahneman, D. (1974). Judgment under Uncertainty: Heuristics and Biases. *Science*.

Gärdenfors, P. (2000). *Conceptual Spaces: The Geometry of Thought*. MIT Press.

Lakoff, G. (1987). *Women, Fire, and Dangerous Things*. University of Chicago Press.

2.2 Ağ Bilimi ve Graf Kuramı

Ağ bilimi, karmaşık sistemleri düğümler ve bağlantılar üzerinden inceleyen disiplinler arası bir alandır. Bu yaklaşım, fizik, matematik, bilgisayar bilimi, sosyoloji ve biyoloji gibi birçok alanın kesişim noktasında yer alır. Temel amacı, çok sayıda etkileşim içeren sistemlerde ortaya çıkan yapısal örüntüleri anlamaktır.

Graf kuramı ise bu alanın matematiksel temelini oluşturur. Bir graf, düğümler ve bu düğümleri birbirine bağlayan bağlantılardan oluşan soyut bir yapıdır. Bu basit tanım, aslında son derece güçlü bir modelleme aracıdır. Çünkü birçok gerçek dünya sistemi bu yapıya indirgenebilir: sosyal ağlar, ulaşım sistemleri, biyolojik etkileşimler ve bilgi sistemleri gibi.

Ağ biliminin en önemli katkılarından biri, karmaşık sistemlerdeki düzeni ortaya çıkarmasıdır. Örneğin küçük dünya ağları, iki rassal düğüm arasında şaşırtıcı derecede kısa bağlantı yolları bulunduğunu gösterir. Ölçekten bağımsız ağlar ise bazı düğümlerin (hub'ların) çok yüksek bağlantı sayısına sahip olduğunu ortaya koyar. Bu tür yapılar, internetten insan sosyal ilişkilerine kadar birçok sistemde gözlemlenir.

Merkezilik kavramı ise ağ içinde hangi düğümlerin daha "önemli" olduğunu ölçmeye yarar. Bu önem, yalnızca bağlantı sayısı ile değil, bağlantıların konumuyla da ilgilidir. Bu nedenle graf kuramı, yalnızca yapıyı değil, yapının içindeki dinamik gücü de analiz eder.

Günümüzde ağ bilimi ve graf kuramı, bilgi grafı sistemlerinin teorik temelini oluşturur. Modern veri sistemleri, yalnızca veri depolamakla kalmaz, aynı zamanda bu verinin içinde saklı olan yapısal ilişkileri ortaya çıkarmaya çalışır. Bu nedenle graf kuramı, dijital dünyanın görünmeyen mimarisini anlamak için vazgeçilmez bir araç haline gelmiştir.

Newman, M. (2010). *Networks: An Introduction*. Oxford University Press.

Barabási, A.-L. (2016). *Network Science*. Cambridge University Press.

Bollobás, B. (1998). *Modern Graph Theory*. Springer.

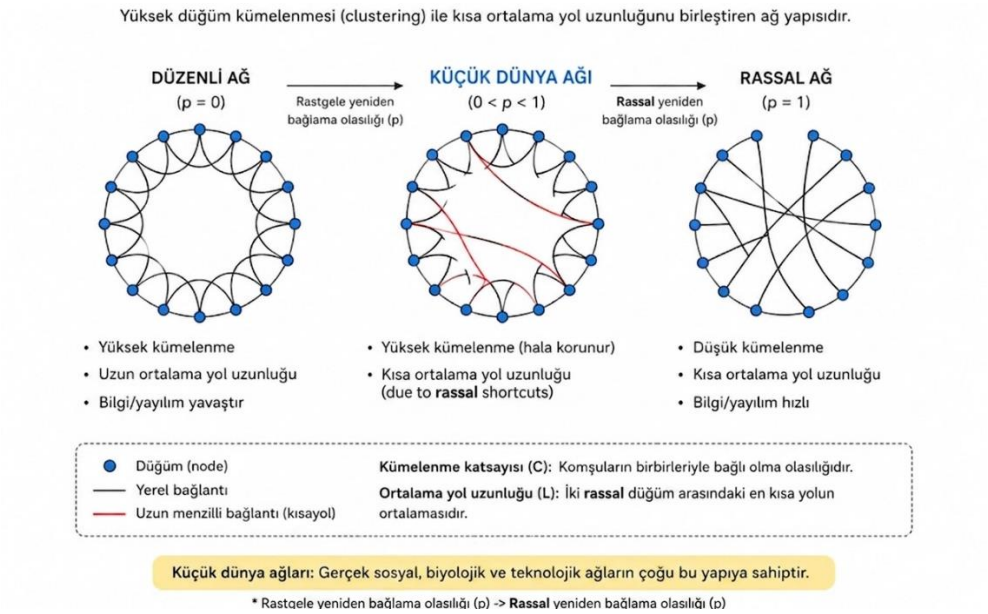
Easley, D., & Kleinberg, J. (2010). *Networks, Crowds, and Markets*. Cambridge University Press.

2.3 Küçük Dünya Ağları

Küçük dünya ağları, karmaşık sistemlerde düğümler arasındaki ortalama mesafenin şaşırtıcı derecede kısa olduğunu gösteren bir ağ yapısıdır. Bu kavram, ilk bakışta sezgisel olmayan bir gerçeği ortaya koyar: Çok büyük ve karmaşık görünen sistemlerde bile, iki rassal nokta arasında genellikle çok az sayıda bağlantı vardır. Bu durum, "altı derece ayrılık" (six degrees of separation) fikriyle de popülerleşmiştir.

Sosyal ilişkiler bu yapının en bilinen örneklerinden biridir. Bir birey, doğrudan tanımadığı bir başka bireye, yalnızca birkaç ara bağlantı üzerinden ulaşabilir. Bu durum yalnızca sosyal ağlarda değil, biyolojik sistemlerde, iletişim ağlarında ve hatta bilgi sistemlerinde de gözlemlenir. Küçük dünya yapısı, yerel kümelenme ile küresel erişilebilirlik arasında bir denge kurar.

Şekil 8. Küçük Dünya Ağı



Şekil 8, ağ biliminin (network science) en temel ve büyüleyici modellerinden biri olan **Duncan Watts ve Steven Strogatz (1998)** modelini, yani **Küçük Dünya Ağları (Small-World Networks)**

teorisini açıklayan son derece net ve öğretici bir infografiktir. Görsel, en üstteki genel tanımıyla "Yüksek düğüm kümelenmesi (clustering) ile kısa ortalama yol uzunluğunu birleştiren ağ yapısıdır" ifadesiyle konuya giriş yapar.

Görselin ana gövdesinde soldan sağa doğru, ağ yapısının "yeniden bağlama olasılığı (p)" parametresine bağlı olarak nasıl evrildiğini gösteren üç farklı dairesel graf modeli yer almaktadır:

1. Düzenli Ağ (Regular Network / p = 0)

Sol başta yer alan ilk dairesel model, tamamen simetrik ve düzenli bir yapıyı temsil eder. Her düğüm sadece en yakınındaki komşularıyla (yerel bağlantılarla) sıkı sıkıya bağlıdır.

- **Yüksek kümelenme:** Komşular birbirine sıkıca bağlıdır.
- **Uzun ortalama yol uzunluğu:** Ağın bir ucundaki düğümden en uzak ucundaki düğüme gitmek için çok sayıda basamak (adım) geçmek gerekir.
- **Bilgi/yayılım yavaştır:** Yapı çok çizgisel ve izole olduğu için akış zaman alır.

2. Küçük Dünya Ağları (Small-World Network / 0 < p < 1)

Ortada yer alan ve teorinin kalbini oluşturan bu modelde, düzenli ağdaki bazı yerel bağlar çok düşük bir olasılıkla (p) koparılarak ağın karşı tarafındaki uzak düğümlere rastgele yeniden bağlanmıştır (görselde kırmızı çizgilerle gösterilen **Uzun menzilli bağlantılar / kısayollar**).

- **Yüksek kümelenme (hala korunur):** Ağ yerel düzeyde hala sıkı dostluklar/bağlar barındırır.
- **Kısa ortalama yol uzunluğu:** Araya atılan o birkaç stratejik kırmızı kısayol (*due to rassal shortcuts*), ağın bir ucundan diğer ucuna olan mesafeyi dramatik bir şekilde kısaltır. Altı derece uzaklık (six degrees of separation) felsefesi tam olarak bu mekanizmaya dayanır.

3. Rassal Ağ (Random Network / p = 1)

Sağ başta yer alan son modelde ise yeniden bağlama olasılığı maksimuma (p=1) ulaşmıştır. Artık yerel hiçbir düzen kalmamış, tüm bağlantılar tamamen rastgele (rassal) bir şekilde dağılmıştır.

- **Düşük kümelenme:** Yakın komşuluklar ve yerel gruplaşmalar tamamen kaybolmuştur.
- **Kısa ortalama yol uzunluğu:** Her yer karmaşık bağlarla örüldüğü için uzaklıklar kısadır.
- **Bilgi/yayılım hızlı:** Ağ tamamen açık bir karmaşa içinde olduğundan akış çok hızlı yayılır ancak düzenli bir yapı yoktur.

Görselin alt kısmındaki kesikli çizgili kutuda, graf teorisindeki iki hayati kavram tanımlanmıştır:

- **Kümelenme katsayısı (C):** Komşuların birbiriyle bağlı olma olasılığını, yani yerel dostlukların sıklığını ölçer.

- **Ortalama yol uzunluğu (L):** İki rassal düğüm arasındaki en kısa yolun ortalamasını, yani ağın ne kadar "küçük" olduğunu gösterir.

En alttaki sarı şeritte yer alan **"Küçük dünya ağları: Gerçek sosyal, biyolojik ve teknolojik ağların çoğu bu yapıya sahiptir"** vurgusu ise bu teorinin neden bu kadar önemli olduğunu açıklar. Gerçek dünyadaki insan toplulukları (sosyal ağlar), beyindeki nöronlar (biyolojik ağlar) ve hatta internet veya elektrik şebekeleri (teknolojik ağlar) ne tamamen düzenli ne de tamamen rastgeledir; onlar tam olarak ortada duran, yani **yüksek kümelenme ile kısa mesafeyi birleştiren birer "Küçük Dünya" ağıdır.**

Bu ağlarda genellikle iki temel özellik birlikte görülür: yüksek kümeleşme (clustering) ve düşük ortalama yol uzunluğu (path length). Yani bireyler kendi çevreleri içinde yoğun bağlantılara sahipken, aynı zamanda ağın genelinde hızlı bir şekilde diğer bölgelere ulaşılabilir. Bu yapı, bilgi yayılımını ve etkileşimi son derece verimli hale getirir.

Bilgi grafı sistemleri açısından küçük dünya yapıları oldukça önemlidir. Çünkü bilgi, bu tür ağlarda hızlı bir şekilde yayılabilir ve ilişkisel bağlantılar üzerinden anlamlı sonuçlar üretilebilir. Modern dijital sistemler, bu yapıyı bilinçli olarak taklit ederek öneri sistemlerinden arama motorlarına kadar birçok uygulamada kullanır.

Watts, D. J., & Strogatz, S. H. (1998). Collective Dynamics of 'Small-World' Networks. *Nature*.

Milgram, S. (1967). The Small World Problem. *Psychology Today*.

Watts, D. J. (2003). *Six Degrees: The Science of a Connected Age*. Norton.

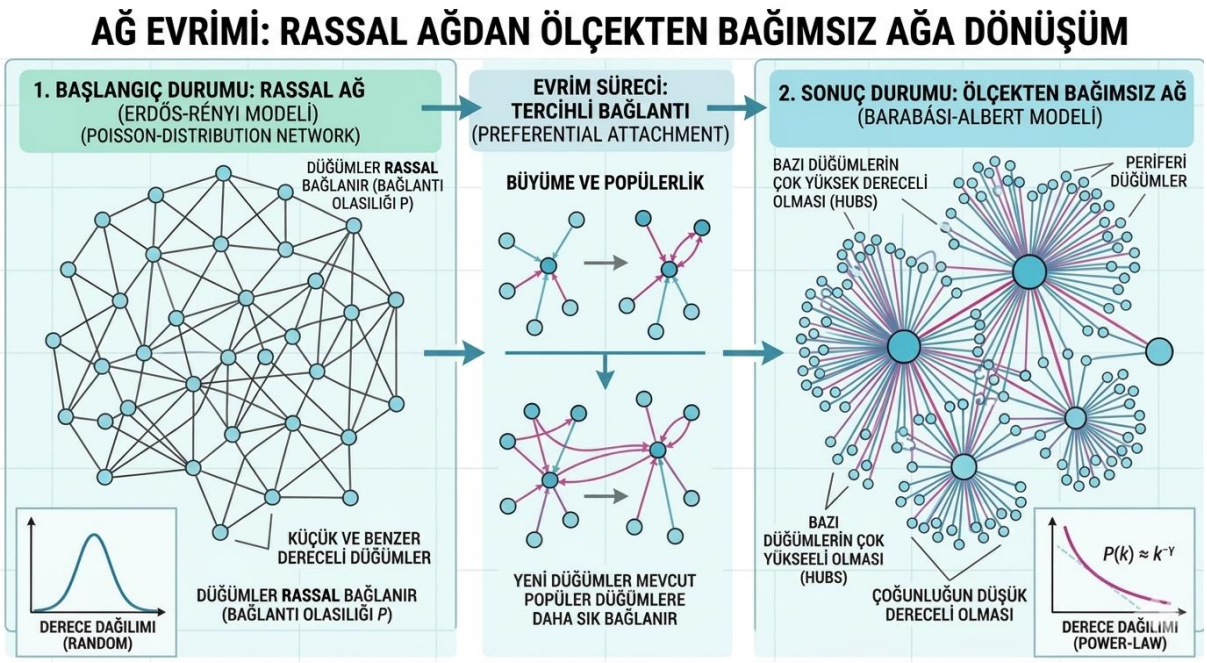
2.4 Ölçekten Bağımsız Ağlar

Ölçekten bağımsız ağlar, ağ içindeki düğüm bağlantılarının eşit dağılmadığı, bunun yerine birkaç düğümün (hub) çok yüksek bağlantıya sahip olduğu, geri kalan düğümlerin ise çok az bağlantı içerdiği ağ yapılarıdır. Bu tür ağlarda bağlantı dağılımı genellikle güç yasası (power law) ile ifade edilir.

Bu yapı, rassal olmayan bir organizasyon biçimine işaret eder. Örneğin internet, sosyal medya platformları ve akademik atıf ağları bu tür dağılıma sahip sistemlerdir. Bazı düğümler olağanüstü derecede merkezi bir rol oynarken, çoğu düğüm yalnızca sınırlı sayıda bağlantıya sahiptir.

Ölçekten bağımsız ağların en önemli özelliklerinden biri, dayanıklılık ve kırılganlık arasındaki paradokstur. Bu tür ağlar rassal düğüm kayıplarına karşı oldukça dayanıklıdır; çünkü çoğu düğüm düşük bağlantılıdır. Ancak yüksek bağlantılı hub'ların hedef alınması durumunda ağın bütün yapısı hızla çökebilir. Bu durum, hem teknik sistemler hem de sosyal yapılar açısından kritik bir analiz alanı oluşturur.

Şekil 9. Ölçekten Bağımsız Ağ



Şekil 9, ağ biliminde (network science) statik ve rastgele modellerden dinamik ve gerçekçi modellere geçişi simgeleyen **"Ağ Evrimi: Rassal Ağdan Ölçekten Bağımsız Ağa Dönüşüm"** sürecini özetlemektedir. Görsel, soldan sağa doğru bir evrim sürecini ve bu sürecin arkasındaki matematiksel mekanizmayı üç ana aşamada ele alır:

1. Başlangıç Durumu: Rassal Ağ (Erdős-Rényi Modeli)

Sol panelde, ağ biliminin klasik teorilerinden biri olan **Erdős-Rényi (Rassal Ağ)** modeli gösterilmektedir.

- **Bağlantı Mantığı:** Düğümler birbirine tamamen rastgele (belirli bir p olasılığı ile) bağlanır.
- **Yapısal Özellik:** Ağdaki tüm düğümler neredeyse **küçük ve benzer derecelidir**. Sistemde aşırı öne çıkan, devasa bağlantılara sahip ayrıcalıklı yapılar yoktur.
- **Derece Dağılımı:** Sol alttaki grafikte görüldüğü gibi, derece dağılımı bir **Poisson Dağılımı** (çan eğrisi benzeri) takip eder. Yani düğümlerin büyük çoğunluğu ağın ortalama derece değerine sahipken, çok az veya çok fazla bağlantısı olan düğümlerin sayısı ihmal edilebilir düzeydedir.

2. Evrim Süreci: Tercihli Bağlantı (Preferential Attachment)

Merkezdeki dikey şerit, ağın statik yapısını bozup onu gerçek dünyadaki ağlara dönüştüren dinamik motoru, yani **"Tercihli Bağlantı"** mekanizmasını açıklar.

- **Zengin Daha Da Zenginleşir (Rich-get-richer):** Ağ zamanla büyürken sisteme eklenen **yeni düğümler, mevcut popüler (derecesi zaten yüksek olan) düğümlere bağlanmayı**

daha sık tercih eder. * Görseldeki küçük şemada, ortadaki popüler düğümün zamanla etrafına nasıl daha fazla bağlantı çekerek bir çekim merkezine dönüştüğü "Büyüme ve Popülerlik" başlığı altında mikro düzeyde modellenmiştir.

3. Sonuç Durumu: Ölçekten Bağımsız Ağ (Barabási-Albert Modeli)

Sağ panelde ise tercihli bağlantı sürecinin nihai sonucu olan ve gerçek dünya ağlarını (İnternet web siteleri, sosyal ağlar, akademik atıf ağları gibi) kusursuz temsil eden **Barabási-Albert (Ölçekten Bağımsız / Scale-Free)** modeli yer alır.

- **Hubs (Merkezler):** Tercihli bağlantı nedeniyle **bazı düğümler çok yüksek dereceli (Hubs)** hale gelmiştir. Görselde büyük mavi daireler ve onlardan fışkıran yüzlerce çizgi bu merkezleri net bir şekilde gösterir.
- **Çoğunluğun Durumu:** Ağın ezici bir çoğunluğu ise sadece birkaç bağlantıya sahip **düşük dereceli periferi (çevre) düğümlerden** oluşur.
- **Derece Dağılımı:** Sağ alttaki grafikte görüldüğü gibi, derece dağılımı artık çan eğrisi değil, bir **Güç Yasası (Power-Law)** grafiğidir ($P(k) \approx k^{-\gamma}$). Bu eğri, sistemde ölçeği belirleyen tek bir ortalama değer olmadığını, çok az sayıda devasa merkezin (hub) ve çok sayıda küçük düğümün bir arada yaşadığı asimetrik bir evreni doğrular.

Özetle bu görsel; eşitlikçi ve rastgele doğan bir sistemin (Rassal Ağ), "popüler olanı seçme" dürtüsüyle (Tercihli Bağlantı) zaman içinde nasıl merkeziyetçi, hiyerarşik ve organik bir güç odağı yapısına (Ölçekten Bağımsız Ağ) evrildiğini teknik ve görsel olarak mükemmel bir biçimde açıklamaktadır.

Bilgi grafı perspektifinden bakıldığında, ölçekten bağımsız yapıların doğal olarak ortaya çıktığı görülür. Bazı varlıklar (örneğin önemli kişiler, büyük kurumlar veya merkezi kavramlar) çok sayıda ilişkiye sahipken, çoğu varlık daha sınırlı bağlantılar içerir. Bu durum, bilginin eşit dağılmadığını, aksine merkezileştiğini gösterir.

Modern yapay zekâ ve büyük ölçekli veri sistemleri, bu tür ağ yapılarını analiz ederek kritik düğümleri, bilgi akışının merkezlerini ve sistemdeki kırılgan noktaları tespit edebilir. Bu nedenle ölçekten bağımsız ağlar, yalnızca teorik bir model değil, aynı zamanda pratik bir analiz aracıdır.

Sonuç olarak, küçük dünya ağları ve ölçekten bağımsız ağlar birlikte düşünüldüğünde, karmaşık sistemlerin hem yerel yoğunluk hem de küresel dengesizlik içeren yapılar olduğu görülür. Bu iki özellik, modern bilgi sistemlerinin ve bilgi grafı mimarilerinin temel davranış kalıplarını anlamak için kritik öneme sahiptir.

Barabási, A.-L., & Albert, R. (1999). Emergence of Scaling in Random Networks. *Science*.

Newman, M. (2005). Power Laws, Pareto Distributions and Zipf's Law. *Contemporary Physics*.

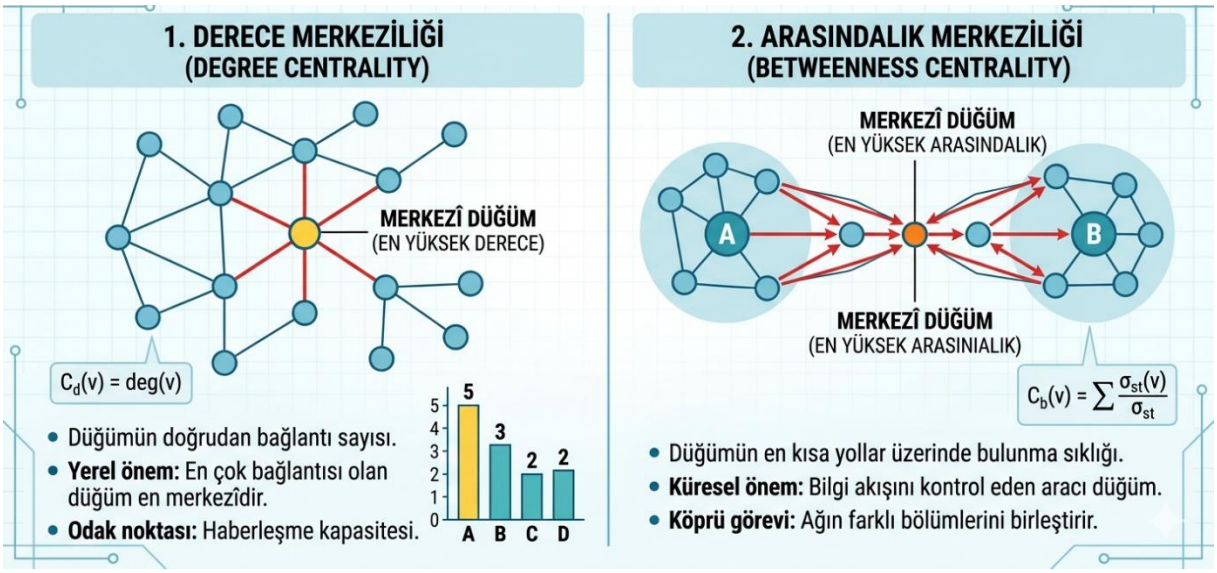
Clauset, A., Shalizi, C. R., & Newman, M. (2009). Power-Law Distributions in Empirical Data. *SIAM Review*.

2.5 Merkezilik Kavramı

Ağ bilimi içinde en kritik kavramlardan biri merkezilik kavramıdır. Merkezilik, bir düğümün ağ içindeki "önemini" ölçmeye yarayan matematiksel ve yapısal bir göstergedir. Ancak burada "önem" kelimesi tek boyutlu bir anlam taşımaz; çünkü ağ içinde önemli olmak farklı biçimlerde tanımlanabilir. Bir düğüm çok sayıda bağlantıya sahip olduğu için önemli olabilir, ağın farklı bölgelerini birbirine bağladığı için kritik olabilir ya da en kısa yollar üzerinde bulunduğu için stratejik bir konumda yer alabilir.

Bu nedenle merkezilik kavramı tek bir ölçü değil, bir ölçüler ailesi olarak ele alınır. En basit biçimiyle derece merkeziliği, bir düğümün sahip olduğu bağlantı sayısını ölçer. Sosyal ağlarda bu, bir kişinin kaç kişiyle doğrudan bağlantısı olduğunu ifade eder. Ancak bu ölçü her zaman yeterli değildir, çünkü yalnızca bağlantı sayısı değil, bu bağlantıların yapısal konumu da önemlidir.

Şekil 10. Derece ve Arasındalık Merkeziliği Kavramları



Şekil 10, ağ analizinde (network analysis) düğümlerin sistem içindeki gücünü, önemini ve konumunu ölçmek için kullanılan en temel iki **Merkezilik (Centrality)** ölçütünü karşılaştırmalı olarak açıklayan çok net bir infografiktir. Görsel, dikey bir çizgiyle iki ana panele ayrılmıştır:

1. Derece Merkeziliği (Degree Centrality)

Sol panelde, bir ağdaki en sezgisel ve hesaplaması en kolay güç ölçütü olan **Derece Merkeziliği** ele alınmıştır. Ağın içinde **sarı renkle** boyanmış olan merkezî düğüm, etrafına çok sayıda kırmızı ok/çizgi uzatarak en yüksek dereceye sahip olduğunu belli eder. Sağ alttaki sütun grafiğinde de A düğümünün (sarı sütun) derece skorunun 5 olduğu, diğerlerinin (B, C, D) ise daha düşük kaldığı gösterilmiştir.

2. Arasındalık Merkeziliği (Betweenness Centrality)

Sağ panelde ise ağın genel topolojisi ve bilgi akışı açısından çok daha stratejik bir ölçüt olan **Arasındalık Merkeziliği** açıklanmıştır. Şemada A kümesi (sol taraftaki ağ grubu) ile B kümesi (sağ taraftaki ağ grubu) arasında iki farklı topluluk yer almaktadır. **Turuncu renkle** gösterilen merkezî düğüm, bu iki büyük topluluğun tam ortasında, adeta tek bir geçiş kapısı gibi konumlanmıştır. A'dan B'ye gitmek isteyen tüm bilgi akışı (kırmızı oklar) bu turuncu düğümden geçmek zorundadır. Görselde küçük bir yazım hatasıyla "EN YÜKSEK ARASINIALIK" yazsa da kastedilen en yüksek arasındalıktır.

Anlamı ve Odak Noktası:

- **Küresel Önem:** Bu düğümün derece sayısı (bağlantı sayısı) sol paneldeki sarı düğüm kadar çok olmayabilir. Ancak konumu gereği **küresel bilgi akışını kontrol eden bir aracı/darboğaz** rolündedir.
- **Köprü Görevi:** Ağın birbirinden kopuk farklı bölümlerini birbirine bağlar. Bu düğüm sistemden çıkarılırsa, ağ iki büyük parçaya bölünebilir veya iletişim tamamen felç olabilir (Örn: Lojistikte kilit bir aktarma limanı veya şirketler arası köprü kuran kilit bir yönetici).

Sol taraftaki **Derece Merkeziliği** bize ağın en popüler, en çok bağlantıya sahip *"yerel liderini"* gösterirken; sağ taraftaki **Arasındalık Merkeziliği** bize ağın farklı dünyaları birbirine bağlayan, bilgi akışını elinde tutan en stratejik *"köprü/aracı aktörünü"* gösterir.

Arasındalık merkeziliği, bir düğümün ağ içindeki bilgi akışında ne kadar “köprü” rolü oynadığını ölçer. Eğer bir düğüm birçok farklı bölge arasındaki en kısa yollar üzerinde yer alıyorsa, bu düğüm stratejik olarak oldukça önemlidir. Yakınlık merkeziliği ise bir düğümün diğer tüm düğümlere ortalama ne kadar kısa mesafede olduğunu ölçer.

Bilgi grafı sistemlerinde merkezilik kavramı özellikle önemlidir. Çünkü bazı varlıklar doğal olarak daha merkezi konumdadır. Örneğin büyük kurumlar, önemli bilim insanları veya temel kavramlar, ağ içinde çok sayıda bağlantıya sahiptir. Bu durum, bilginin eşit dağılmadığını, aksine belirli düğümler etrafında yoğunlaştığını gösterir.

Modern yapay zekâ sistemleri ve arama motorları, merkezilik ölçülerini kullanarak hangi bilgilerin daha “önemli” veya “ilişkisel olarak güçlü” olduğunu belirleyebilir. Bu nedenle

merkezilik, yalnızca matematiksel bir kavram değil, aynı zamanda bilgi sistemlerinde anlamın dağılımını açıklayan temel bir araçtır.

Freeman, L. C. (1979). Centrality in Social Networks. *Social Networks*.

Borgatti, S. P. (2005). Centrality and Network Flow. *Social Networks*.

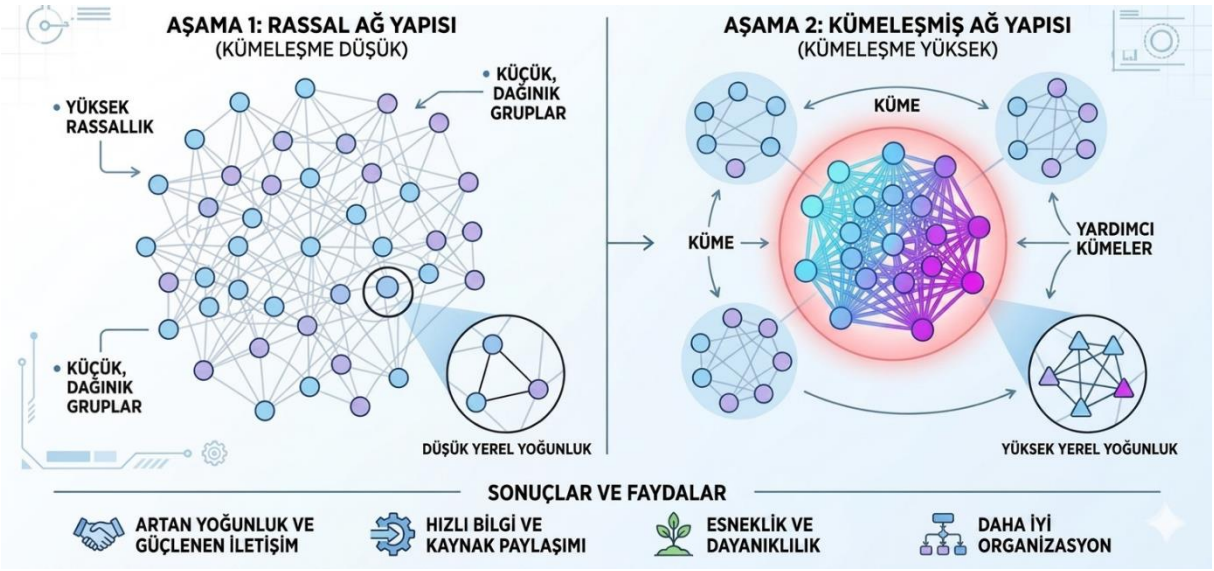
Jackson, M. O. (2008). *Social and Economic Networks*. Princeton University Press.

2.6 Topluluklar ve Kümeleşme

Ağ yapılarının bir diğer temel özelliği topluluklar (communities) ve kümeleşme (clustering) kavramlarıdır. Gerçek dünya ağları genellikle homojen değildir; bunun yerine belirli bölgelerde yoğun bağlantıların olduğu, diğer bölgelerden ise kısmen ayrışmış alt yapılar içerir. Bu alt yapılar, topluluklar veya kümeler olarak adlandırılır.

Kümeleşme, bir düğümün komşularının birbirleriyle ne kadar bağlantılı olduğunu ölçer. Eğer bir kişinin arkadaşları da kendi aralarında yüksek oranda bağlantılıysa, bu kişi yüksek kümeleşme katsayısına sahip bir ağ parçasında yer alır. Bu durum, sosyal ağlarda “arkadaş grupları” veya “sosyal çevreler” olarak gözlemlenir.

Şekil 11. Kümeleşme



Şekil 11, bir ağ yapısının zaman içindeki organizasyonunu ve yapısal olgunlaşmasını gösteren **"Ağ Yapılarında Kümelenme ve Yoğunluk Evrimi"** temalı akademik bir infografiktir. Görsel, düzensiz ve dağınık bir sistemin (Aşama 1), belirli dinamiklerle nasıl daha verimli, modüler ve güçlü gruplara (Aşama 2) dönüştüğünü iki ana panelde karşılaştırmalı olarak ele almaktadır:

Aşama 1: Rassal Ağ Yapısı (Kümelenme Düşük)

Sol taraftaki panel, sistemin henüz organize olmadığı, bağların rastgele dağıldığı ilk evreyi temsil eder.

- **Yüksek Rassallık:** Düğümler arasındaki bağlantılar belirli bir amaca veya topluluk mantığına göre değil, rastgele (stochastic) örülmüştür. Bu yüzden ağın geneline bakıldığında homojen fakat karmaşık bir "örümcek ağı" görüntüsü hakimdir.
- **Küçük ve Dağınık Gruplar:** Varlıklar kendi içlerinde güçlü yerel ittifaklar kuramamıştır. Sistem, birbirinden bağımsız hareket eden küçük ve dağınık gruplardan oluşur.
- **Düşük Yerel Yoğunluk:** Paneldeki büyüteç içine alınmış mikro şemada görüldüğü gibi, komşu düğümlerin birbirleriyle bağ kurma oranı çok düşüktür. Üç düğüm çizgisel bir hatla bağlıdır ancak bir üçgen oluşturacak şekilde kenetlenmemiştir. Bu durum, yerel güven ve bilgi koruma mekanizmalarının zayıf olduğunu gösterir.

Aşama 2: Kümeleşmiş Ağ Yapısı (Kümelenme Yüksek)

Sağ taraftaki panel, aynı düğüm kümesinin mantıksal bir evrim geçirerek **modüler bir topluluk yapısına (community structure)** dönüştüğü olgunlaşma evresini gösterir.

- **Küme (Merkezî Topluluk):** Panelinin merkezinde, parıldayan kırmızı bir dairesel alan içinde çok yüksek yoğunluklu, adeta birbirine kenetlenmiş (clique benzeri) devasa bir ana küme oluşmuştur. Burası sistemin çekirdeğidir.
- **Yardımcı Kümeler (Periferi Modüller):** Ana merkezî kümenin etrafında, onunla çift yönlü oklarla haberleşen, kendi içlerinde de organize olmuş daha küçük "Yardımcı Kümeler" yer almaktadır. Bu durum, ağın ölçeklenebilir ve hiyerarşik bir uzmanlaşma kazandığını gösterir.
- **Yüksek Yerel Yoğunluk:** Sağ alttaki büyüteç içinde, Aşama 1'deki dağınık yapının aksine, düğümlerin kendi aralarında çok sayıda çapraz bağ kurarak tam bir "üçgen/klik" matrisi (tam bağlı graf yapısı) oluşturduğu görülmektedir. Bu yüksek yerel yoğunluk, ağ biliminde yüksek kümelenme katsayısına (C) karşılık gelir.

Görselin en alt bandında, ağın rassal yapıdan kümeleşmiş yapıya geçmesiyle sistemin kazandığı 4 büyük stratejik avantaj ikonlarla özetlenmiştir:

1. **Artan Yoğunluk ve Güçlenen İletişim:** Sıkı bağlar sayesinde düğümler arası etkileşim maksimuma çıkar.
2. **Hızlı Bilgi ve Kaynak Paylaşımı:** Kümelerin kendi içindeki yüksek yoğunluk, enformasyonun tıkanmadan, çok hızlı yayılmasını sağlar.
3. **Esneklik ve Dayanıklılık:** Ağ modüler hale geldiği için, kümelerden biri zarar görse bile diğer yardımcı kümeler kendi iç bağları sayesinde ayakta kalır ve sistem çökmez.
4. **Daha İyi Organizasyon:** Kaotik bir karmaşadan, yönetilebilir ve işlevsel bir ekosistem mimarisine geçiş sağlanmış olur.

Topluluk tespiti (community detection) ise ağ içinde doğal olarak oluşmuş grupları ortaya çıkarmayı amaçlar. Bu gruplar çoğu zaman önceden tanımlanmamıştır; verinin yapısından

otomatik olarak çıkarılır. Örneğin bir akademik ağda belirli araştırma alanları etrafında kümelenmiş bilim insanları, doğal topluluklar oluşturur.

Bilgi grafı açısından topluluklar oldukça önemlidir. Çünkü bilgi genellikle rassal dağılmaz; belirli kavramsal alanlar etrafında kümelenir. Örneğin "yapay zekâ", "biyoinformatik" veya "finans" gibi alanlar kendi içinde yoğun bağlantılar içerir, ancak diğer alanlarla daha zayıf bağlantılara sahiptir.

Kümeleşme yapıları aynı zamanda bilginin nasıl organize edildiğini de gösterir. Yüksek kümeleşme, yerel bilgi yoğunluğunu artırırken, farklı kümeler arasındaki bağlantılar yeni keşiflerin ortaya çıkmasını sağlar. Bu nedenle yenilik çoğu zaman kümeler arası köprülerde ortaya çıkar.

Modern graf veri tabanı sistemleri ve yapay zekâ uygulamaları, topluluk tespiti algoritmaları kullanarak veri içindeki gizli yapıları ortaya çıkarır. Bu yapı, öneri sistemlerinden bilimsel keşif analizlerine kadar birçok alanda kullanılır.

Sonuç olarak merkezilik ve topluluk yapıları birlikte düşünüldüğünde, ağların yalnızca bağlantılardan oluşan yapılar olmadığı, aynı zamanda hiyerarşik, kümelenmiş ve farklı önem seviyelerine sahip düğümlerden oluşan karmaşık sistemler olduğu görülür. Bu da bilgi grafı yaklaşımının neden bu kadar güçlü olduğunu açıklar: çünkü gerçek dünyanın yapısı zaten bir graf yapısıdır.

Girvan, M., & Newman, M. E. J. (2002). Community Structure in Social and Biological Networks. *PNAS*.

Fortunato, S. (2010). Community Detection in Graphs. *Physics Reports*.

Blondel, V. D. et al. (2008). Fast Unfolding of Communities in Large Networks. *JSTAT*.

2.7 Bilgi Akışı ve Ağ Dinamikleri

Ağlar yalnızca statik yapılardan ibaret değildir; aslında sürekli değişen, evrilen ve zaman içinde yeniden şekillenen dinamik sistemlerdir. Bu nedenle ağ bilimi içinde en önemli konulardan biri bilgi akışı ve ağ dinamikleri konusudur. Bir ağın yapısı sabit olsa bile, o ağ üzerinde dolaşan bilgi sürekli hareket halindedir.

Bilgi akışı, bir düğümden diğerine geçen enformasyonun nasıl yayıldığını inceler. Bu yayılım bazen bir sosyal medya gönderisi, bazen bir hastalık, bazen de bir fikir olabilir. Özellikle sosyal ağlarda bu süreç oldukça hızlıdır ve çoğu zaman doğrusal değildir. Bir bilgi parçası yalnızca bir kaynaktan tüm ağa yayılmaz; bunun yerine çoklu kaynaklar, tekrar eden etkileşimler ve geri besleme döngüleri içerir.

Ağ dinamikleri ise ağın yapısal olarak zaman içinde nasıl değiştiğini inceler. Yeni düğümler eklenir, eski bağlantılar zayıflar, bazı ilişkiler güçlenir. Örneğin sosyal ağlarda insanlar yeni

bağlantılar kurar, bazı ilişkiler zamanla kaybolur. Akademik ağlarda ise yeni makaleler eski çalışmalara atıf yaparak yapıyı sürekli yeniden şekillendirir.

Bilgi grafı perspektifinden bakıldığında bu dinamik yapı son derece kritiktir. Çünkü bilgi sabit değildir; sürekli güncellenir, genişler ve yeniden organize olur. Bu nedenle modern bilgi sistemleri yalnızca veriyi saklamakla kalmaz, aynı zamanda değişimi de takip eder.

Özellikle yapay zekâ sistemlerinde bilgi akışının kontrolü, doğruluk ve güncellik açısından büyük önem taşır. Yanlış veya eski bilgi, ağ içinde hızla yayılabilir ve sistemin genel davranışını etkileyebilir. Bu nedenle bilgi akışını anlamak, sadece teknik değil aynı zamanda epistemolojik bir meseledir.

Granovetter, M. (1973). The Strength of Weak Ties. *American Journal of Sociology*.

Kempe, D., Kleinberg, J., & Tardos, É. (2003). Maximizing the Spread of Influence.

Watts, D. J. (2002). A Simple Model of Global Cascades on Random Networks.

2.8 Sosyal Ağlardan Bilgi Ağlarına

Sosyal ağlar, graf düşüncenin en doğal ve gözlemlenebilir örneklerinden biridir. İnsanlar arasındaki ilişkiler, etkileşimler ve iletişimler bir ağ yapısı oluşturur. Bu ağlarda düğümler bireyleri, bağlantılar ise sosyal ilişkileri temsil eder. Ancak bu yapı yalnızca sosyal etkileşimleri değil, aynı zamanda bilgi üretimini ve yayılımını da içerir.

Zaman içinde sosyal ağların analizi, daha geniş bir kavramsal dönüşüme yol açmıştır: sosyal ağlardan bilgi ağlarına (geçiş. Bu geçiş, yalnızca insanlar arasındaki ilişkileri değil, aynı zamanda kavramlar, belgeler, olaylar ve fikirler arasındaki ilişkileri de modellemeyi mümkün kılmıştır.

Örneğin akademik dünyada atıf ağları, makaleler arasındaki bilgi akışını gösterir. Bir makale başka bir makaleye atıf yaptığında, aslında bilgi bir düğümden diğerine aktarılmış olur. Bu yapı, bilginin nasıl evrildiğini ve hangi fikirlerin merkezde yer aldığını anlamak için güçlü bir araçtır.

Benzer şekilde internet üzerindeki bağlantılar da bir bilgi ağı oluşturur. Web sayfaları yalnızca içerik değil, aynı zamanda birbirine referans veren düğümlerdir. Bu yapı, arama motorlarının temel çalışma prensibini oluşturur.

Bilgi grafı yaklaşımı, bu dönüşümün en ileri aşamasıdır. Artık yalnızca insanlar değil, kavramlar, nesneler ve olaylar da bir ağın parçası olarak modellenir. Örneğin bir şirket, çalışanları, ürünleri ve müşterileriyle birlikte tek bir ilişkisel sistem içinde temsil edilebilir.

Bu dönüşümün en önemli sonucu şudur: bilgi artık izole birimler halinde değil, ilişkiler ağı içinde anlam kazanmaktadır. Bu nedenle bilgi sistemleri, sosyal ağlardan öğrenerek daha geniş bir “anlam ağı” modeline evrilmiştir.

Castells, M. (2010). *The Rise of the Network Society*. Wiley-Blackwell.

Boyd, D., & Ellison, N. B. (2007). Social Network Sites: Definition, History, and Scholarship. *JCMC*.

Easley, D., & Kleinberg, J. (2010). *Networks, Crowds, and Markets*. Cambridge University Press.

2.9 Graf Düşünme: Dünyayı İlişkilerle Okumak

Graf düşünme, dünyayı nesnelerden ziyade ilişkiler üzerinden anlamlandırma biçimidir. Bu yaklaşımda temel soru "Bu nedir?" değil, "Bu neyle bağlantılıdır?" olur. Böylece gerçeklik, sabit kategoriler yerine dinamik bir ilişkiler ağı olarak görülür.

Bu düşünme biçimi, modern bilimsel ve teknolojik sistemlerin temelini oluşturur. Çünkü karmaşık sistemler, yalnızca parçaların özellikleriyle değil, parçalar arasındaki ilişkilerle açıklanabilir. Bir ekosistem, bir ekonomi ya da bir sosyal yapı, tekil unsurların toplamı değil, bu unsurlar arasındaki etkileşimlerin sonucudur.

Graf düşünce, aynı zamanda bilgi grafı sistemlerinin felsefi temelidir. Bir sistemde varlıklar (entities) ve ilişkiler (relations) birlikte anlam üretir. Bu nedenle modern yapay zekâ ve veri sistemleri, dünyayı bir graf olarak modellemeye giderek daha fazla yönelmektedir.

Bu yaklaşım, geleneksel doğrusal düşünme biçiminden önemli bir kopuşu temsil eder. Artık bilgi, sıralı listeler veya kategoriler halinde değil, ağ yapıları içinde organize edilir. Bu da daha esnek, daha ölçeklenebilir ve daha anlamlı bilgi modelleri üretir.

Graf düşüncenin en güçlü yönlerinden biri, gizli bağlantıları görünür hale getirmesidir. Farklı alanlar arasında beklenmedik ilişkiler ortaya çıkarılabilir. Örneğin biyoloji ile bilgisayar bilimi, ekonomi ile sosyal ağ analizi veya tarih ile veri bilimi arasında beklenmedik bağlantılar keşfedilebilir.

Sonuç olarak graf düşünce, yalnızca bir teknik araç değil, aynı zamanda bir zihinsel modeldir. Dünyayı ilişkiler ağı olarak okumak, hem bilimsel keşifleri hem de yapay zekâ sistemlerini daha derin bir anlayış seviyesine taşır. Bu nedenle modern çağın en önemli bilişsel dönüşümlerinden biri, doğrusal düşünceden graf düşünceye geçiştir.

Barabási, A.-L. (2016). *Network Science*. Cambridge University Press.

Sowa, J. F. (2000). *Knowledge Representation*. Brooks/Cole.

Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach*. Pearson.

Hogan, A. et al. (2021). Knowledge Graphs. *ACM Computing Surveys*.

BÖLÜM 3 — BİLGİ GRAFI MiMARiSi

Bu bölümde bilgi grafı sistemlerinin nasıl inşa edildiği, hangi katmanlardan oluştuğu ve bu katmanların bir araya gelerek nasıl çalışan bir mimari oluşturduğu ele alınacaktır. Önceki bölümlerde bilgi grafının ne olduğu ve neden önemli olduğu tartışılmıştı. Bu bölümde ise işin "nasıl" kısmına, yani mühendislik tarafına geçiyoruz.

Bilgi grafı mimarisi, yalnızca veriyi depolayan bir yapı değildir; aynı zamanda veriyi anlamlı hale getiren, ilişkileri kuran, tutarlılığı sağlayan ve gerektiğinde bu bilgiyi sorgulanabilir hale getiren çok katmanlı bir sistemdir. Bu nedenle bir bilgi grafı, tek bir yazılım bileşeni değil, birden fazla bileşenin birlikte çalıştığı bir ekosistem olarak düşünülmelidir.

Bu mimarinin merkezinde varlıklar ve bu varlıklar arasındaki ilişkiler yer alır. Ancak bu ilişkilerin doğru şekilde kurulabilmesi için önce verinin nasıl temsil edileceğinin belirlenmesi gerekir. Bu noktada üçlü yapı (özne–yüklem–nesne) temel bir rol oynar. Ardından şema tasarımı ile sistemin iskeleti oluşturulur ve ontoloji geliştirme süreci ile kavramsal çerçeve tanımlanır.

Gerçek dünyadan gelen veriler çoğu zaman dağınık, eksik ve tutarsızdır. Bu nedenle varlık eşleştirme problemi ve veri bütünleştirme süreçleri kritik hale gelir. Farklı kaynaklardan gelen aynı varlığın birleştirilmesi, bilgi grafının doğruluğu açısından belirleyicidir.

Bunun üzerine inşa edilen üstveri (metadata) ve anlamsal katman, sistemin yalnızca veriyi değil, verinin bağlamını da anlamasını sağlar. Bu katman, bilgi grafını basit bir veri yapısından çıkarıp anlamsal bir modele dönüştürür.

Tüm bu bileşenler, bir bilgi grafı işlem hattı içinde bir araya gelir. Bu hat, verinin alınmasından işlenmesine, depolanmasından sorgulanmasına kadar tüm süreci kapsar. Daha ileri sistemlerde bu yapı gerçek zamanlı bilgi grafı sistemlerine dönüşerek sürekli güncellenen veri akışlarını yönetebilir.

Son aşamada ise graf destekli sorgu ile üretim mimarileri (GrafRAG) devreye girer. Bu yaklaşım, bilgi grafı ile büyük dil modellerini birleştirerek hem doğru hem de bağlamsal olarak zengin yanıtlar üretmeyi hedefler. Bu bölüm boyunca, bilgi grafı mimarisinin yalnızca teknik bir yapı değil, aynı zamanda modern yapay zekâ sistemlerinin temel düşünme biçimlerinden biri olduğu görülecektir.

3.1 Üçlü Yapılar: Özne–Yüklem–Nesne

Bilgi grafı mimarisinin en temel yapı taşı üçlü yapılardır. Bu yapı, Özne–Yüklem–Nesne (Subject–Predicate–Object) formu ile bilginin en küçük anlamlı birimini temsil eder. Bu yaklaşım, veriyi yalnızca saklanacak bir kayıt olarak değil, anlam taşıyan bir ilişki ifadesi olarak ele alır. Her üçlü yapı, iki varlık arasındaki bir ilişkiyi tanımlar ve bu ilişki sayesinde bilgi, ağ içinde dolaşabilir hale gelir.

Özne (subject), hakkında bilgi verilen varlığı temsil eder. Bu bir kişi, kurum, olay ya da soyut bir kavram olabilir. Nesne (object), bu ilişkinin hedefini ifade eder. Yüklem (predicate) ise özne ile nesne arasındaki ilişki türünü tanımlar. Örneğin "Einstein fizikçidir" ifadesi bir üçlü yapıdır: Einstein (özne), fizikçidir (yüklem), fizikçi (nesne). Bu basit yapı, daha karmaşık bilgi sistemlerinin temelini oluşturur.

Bu modelin en önemli avantajı, bilginin atomik hale getirilmesidir. Yani büyük ve karmaşık ifadeler daha küçük, yeniden kullanılabilir ilişki parçalarına ayrılır. Bu sayede aynı varlık farklı bağlamlarda tekrar kullanılabilir ve bilgi grafı içinde farklı ilişkilerle bağlanabilir. Örneğin Einstein hem "fizikçidir", hem "Nobel ödülü kazanmıştır", hem de "Görelilik teorisini geliştirmiştir" gibi birçok üçlü yapının öznesi olabilir.

Üçlü yapıların gücü, esnekliğinden gelir. Geleneksel veri tabanlarında tablo şemaları önceden sıkı bir şekilde tanımlanırken, üçlü yapılar daha dinamik bir modelleme imkânı sunar. Yeni bir ilişki eklemek için tüm yapıyı değiştirmek gerekmez; yalnızca yeni bir üçlü eklemek yeterlidir. Bu özellik, bilgi grafı sistemlerini özellikle değişken ve büyüyen veri yapıları için ideal hale getirir.

Ayrıca üçlü yapılar, Semantik Web yaklaşımının da temelini oluşturur. Bu yaklaşımda amaç, makinelerin yalnızca veriyi saklaması değil, aynı zamanda verinin anlamını da "okuyabilmesidir". Üçlü yapı, bu anlam okuma sürecinin en küçük birimidir. RDF (Resource Description Framework) gibi standartlar da bu mantık üzerine kurulmuştur.

Bilgi grafı sistemlerinde üçlü yapılar sadece veri temsil aracı değil, aynı zamanda sorgulanabilir anlam birimleridir. Bu sayede "kim neyi ne zaman yaptı?" gibi karmaşık sorular, graf üzerinde dolaşarak yanıtlanabilir hale gelir. Bu yapı, klasik SQL mantığından farklı olarak, ilişkiler üzerinden düşünmeyi zorunlu kılar.

Sonuç olarak üçlü yapılar, bilgi grafı mimarisinin atomlarıdır. Nasıl ki fizik dünyasında atomlar birleşerek molekülleri oluşturuyorsa, bilgi grafı dünyasında da üçlü yapılar birleşerek karmaşık bilgi ağlarını oluşturur. Bu nedenle üçlü yapı mantığını anlamak, graf tabanlı düşünmenin en kritik adımlarından biridir.

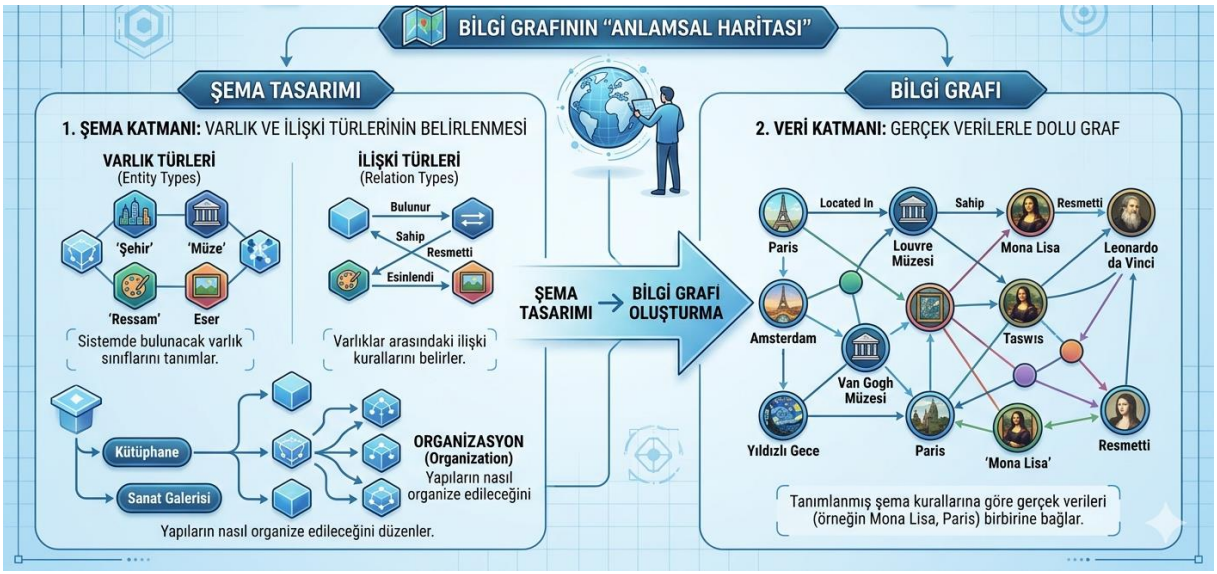
3.2 Şema Tasarımı

Şema tasarımı, bir bilgi grafının iskeletini oluşturan en kritik mimari aşamalardan biridir. Şema, sistemde hangi varlık türlerinin bulunacağını, bu varlıklar arasındaki hangi ilişki türlerinin tanımlanacağını ve bu yapıların nasıl organize edileceğini belirler. Başka bir ifadeyle şema, bilgi grafının "anlamsal haritasıdır".

Geleneksel ilişkisel veri tabanlarında şema tasarımı genellikle katı ve önceden belirlenmiş bir yapıya sahiptir. Tablolar, sütunlar ve veri tipleri baştan tanımlanır ve sistem bu yapıya sıkı sıkıya bağlı kalır. Ancak bilgi grafı sistemlerinde şema daha esnek bir yapıya sahiptir. Bu esneklik, gerçek dünyanın karmaşıklığını daha iyi modelleme imkânı sağlar.

Şema tasarımında ilk adım, varlıkların tanımlanmasıdır. Örneğin bir akademik bilgi grafında "yazar", "makale", "konferans", "kurum" gibi varlık türleri belirlenir. Ardından bu varlıklar arasındaki ilişkiler tanımlanır: "yazar yazar makale", "makale sunulur konferans", "yazar çalışır kurum" gibi.

Şekil 12. Bilgi Grafının Anlamsal Haritası



Şekil 12, bir Bilgi Grafının "Anlamsal Haritası" başlığı altında, graf teknolojilerinde soyut kurallar bütünü (Metaveri/Şema) ile somut gerçeklik (Veri) arasındaki ilişkiyi harika bir şekilde modelleyen akademik bir infografiktir. Görsel, bir bilgi grafı inşa edilirken geçilen iki ana aşamayı soldan sağa doğru bir süreç okuyla birbirine bağlar:

Sol taraftaki büyük panel, bilgi grafının mimari temelini, yani kurallar kitabını (**Ontoloji / Şema**) temsil eder. Henüz sistemde gerçek kişilerin veya nesnelerin isimleri yoktur; sadece kavramsal sınıflar tanımlanır:

- **Varlık Türleri (Entity Types):** Sistemde bulunabilecek nesne sınıfları soyut olarak belirlenir. Şemada; 'Şehir', 'Müze', 'Ressam' ve 'Eser' gibi genel sınıflar (mavi, yeşil, turuncu ikonlar) tanımlanmıştır.
- **İlişki Türleri (Relation Types):** Bu soyut varlıkların birbirleriyle hangi kurallara göre bağlanabileceğini söyler. Örneğin; bir ressamın eseri Resmettiği, bir müzenin bir esere Sahip olduğu veya bir müzenin bir şehirde Bulunur olduğu gibi anlamsal kurallar burada koyulur.

- **Organizasyon (Organization):** Sol altta yer alan hiyerarşik şemada, yapıların nasıl organize edileceği (Örn: Kütüphane veya Sanat Galerisi gibi üst kategorilerin belirlenmesi) ve verinin taksonomik düzeni kurgulanır.

Sağ taraftaki panel, sol tarafta tasarlanan şema kurallarına göre **gerçek dünya verilerinin sisteme yüklendiği ve birbirine bağlandığı** yaşayan katmandır.

- **Somut Düğümler ve Gerçek Bağlar:** Sol taraftaki soyut 'Şehir' kavramı burada Paris ve Amsterdam olarak; 'Müze' kavramı Louvre Müzesi ve Van Gogh Müzesi olarak ete kemiğe bürünmüştür.
- **Anlamsal İlişkilerin İşlemesi:** Tasarlanan kurallar doğrultusunda veriler örümcek ağı gibi birbirine bağlanır:
    - Paris — Located In —► Louvre Müzesi
    - Louvre Müzesi — Sahip —► Mona Lisa
    - Leonardo da Vinci — Resmetti —► Mona Lisa
    - Van Gogh Müzesi — Sahip —► Yıldızlı Gece

Panelin en altındaki açıklama kutusunda belirtildiği gibi, bu katman **"Tanımlanmış şema kurallarına göre gerçek verileri birbirine bağlar."** ### Özetle Süreç (Şema Tasarımı —► Bilgi Grafi Oluşturma) Görselin tam ortasında elinde tabletle dünyayı inceleyen bir insan figürü ve sağa doğru uzanan büyük mavi ok, bu iki katman arasındaki geçişi gösterir. Özetle sol taraf bir veritabanının **"Anayasası" (Blueprint)**, sağ taraf ise o anayasaya göre inşa edilmiş, içinde gerçek insanların, sanat eserlerinin ve şehirlerin yaşadığı "Canlı Şehir/Bilgi Ağı"dır.

Bu süreçte en önemli tasarım kararı, hangi bilginin düğüm olarak, hangi bilginin ilişki olarak temsil edileceğidir. Yanlış bir modelleme kararı, grafın performansını ve anlam gücünü ciddi şekilde etkileyebilir. Örneğin bazı durumlarda bir özellik olarak düşünülebilecek bir bilgi, aslında ayrı bir varlık olarak modellenmelidir.

Şema tasarımında bir diğer önemli konu hiyerarşi ve genelleme ilişkileridir. “Kişi” genel bir varlık türüyken, “öğrenci” veya “akademisyen” onun alt türleri olabilir. Bu tür ontolojik yapılar, grafın anlamsal derinliğini artırır ve sorguların daha güçlü hale gelmesini sağlar.

Modern bilgi grafı sistemlerinde şema tasarımı genellikle statik değildir; zaman içinde evrilir. Yeni veri kaynakları eklendikçe şema genişler ve yeniden düzenlenir. Bu durum, bilgi grafını yaşayan bir sistem haline getirir. Yani şema, sabit bir yapı değil, sürekli gelişen bir organizmadır.

Şema tasarımının başarısı, sistemin hem esnekliğini hem de tutarlılığını doğrudan etkiler. Aşırı katı bir şema sistemin genişlemesini engellerken, aşırı gevşek bir şema ise veri tutarlılığını bozabilir. Bu nedenle şema tasarımı, bilgi grafı mimarisinde denge kurma sanatıdır.

Sonuç olarak şema tasarımı, yalnızca teknik bir modelleme süreci değil, aynı zamanda bilgiyi nasıl anlamlandırdığımızı belirleyen kavramsal bir çerçevedir. İyi tasarlanmış bir şema, veriyi yalnızca depolamaz; onu anlamlı ilişkiler ağına dönüştürür.

3.3 Ontoloji Geliştirme Süreci

Ontoloji geliştirme süreci, bilgi grafı mimarisinin en soyut ama en güçlü katmanlarından biridir. Ontoloji, bir alanın kavramlarını, bu kavramlar arasındaki ilişkileri ve bu ilişkilerin anlamını formel olarak tanımlayan yapıdır. Başka bir ifadeyle ontoloji, bir "bilgi evreninin kurallar sistemi"dir.

Bir ontoloji geliştirme süreci, genellikle alan analizi ile başlar. Öncelikle modellenecek domain (alan) belirlenir: sağlık, finans, akademi veya lojistik gibi. Bu alan içinde hangi temel kavramların bulunduğu, bu kavramların nasıl sınıflandırıldığı ve birbirleriyle nasıl ilişkili olduğu analiz edilir.

İkinci aşama kavramsal modellemedir. Bu aşamada varlık türleri, alt türler ve ilişkiler tanımlanır. Örneğin sağlık alanında "hasta", "hastalık", "ilaç" ve "tedavi" gibi kavramlar belirlenir. Ardından bu kavramlar arasındaki ilişkiler formelleştirilir.

Ontoloji geliştirme sürecinin en önemli yönlerinden biri, tutarlılık kontrolüdür. Aynı kavramın farklı yerlerde farklı anlamlarda kullanılmaması gerekir. Örneğin "hasta" kavramı hem bir kişi hem de bir durum olarak kullanılmamalıdır. Bu tür çakışmalar, ontolojik tutarsızlık olarak adlandırılır.

Üçüncü aşama, formalizasyon sürecidir. Bu aşamada ontoloji, makine tarafından anlaşılabilir bir formata dönüştürülür. RDF, OWL gibi standartlar bu amaçla kullanılır. Bu sayede ontoloji yalnızca insanlar tarafından değil, makineler tarafından da işlenebilir hale gelir.

Ontoloji geliştirme süreci aynı zamanda dinamik bir süreçtir. Yeni bilgiler ortaya çıktıkça ontoloji güncellenir ve genişletilir. Bu durum özellikle hızlı değişen alanlarda (örneğin yapay zekâ veya biyoteknoloji) kritik öneme sahiptir.

Bilgi grafı sistemlerinde ontoloji, verinin nasıl yorumlanacağını belirleyen üst katmandır. Aynı veri, farklı ontolojiler altında farklı anlamlar kazanabilir. Bu nedenle ontoloji, yalnızca teknik bir yapı değil, aynı zamanda anlamsal bir çerçevedir.

Sonuç olarak ontoloji geliştirme süreci, bilgi grafı mimarisinin "anlam katmanını" inşa eder. Şema yapıyı tanımlarken, ontoloji bu yapının ne anlama geldiğini belirler. Bu iki katman birlikte çalışarak, ham veriyi anlamlı bilgiye dönüştürür.

3.4 Varlık Eşleştirme Problemi

Varlık eşleştirme problemi (entity resolution problem), bilgi grafı sistemlerinin en kritik ve aynı zamanda en zor problemlerinden biridir. Temel olarak farklı veri kaynaklarında yer alan aynı gerçek dünya varlığının tespit edilmesi ve tekil bir temsil altında birleştirilmesi sürecini ifade eder. Bu varlık bir kişi, kurum, ürün, olay veya herhangi bir gerçek dünya nesnesi olabilir.

Problem basit görünse de, gerçek dünya verisinin doğası bu süreci son derece karmaşık hale getirir.

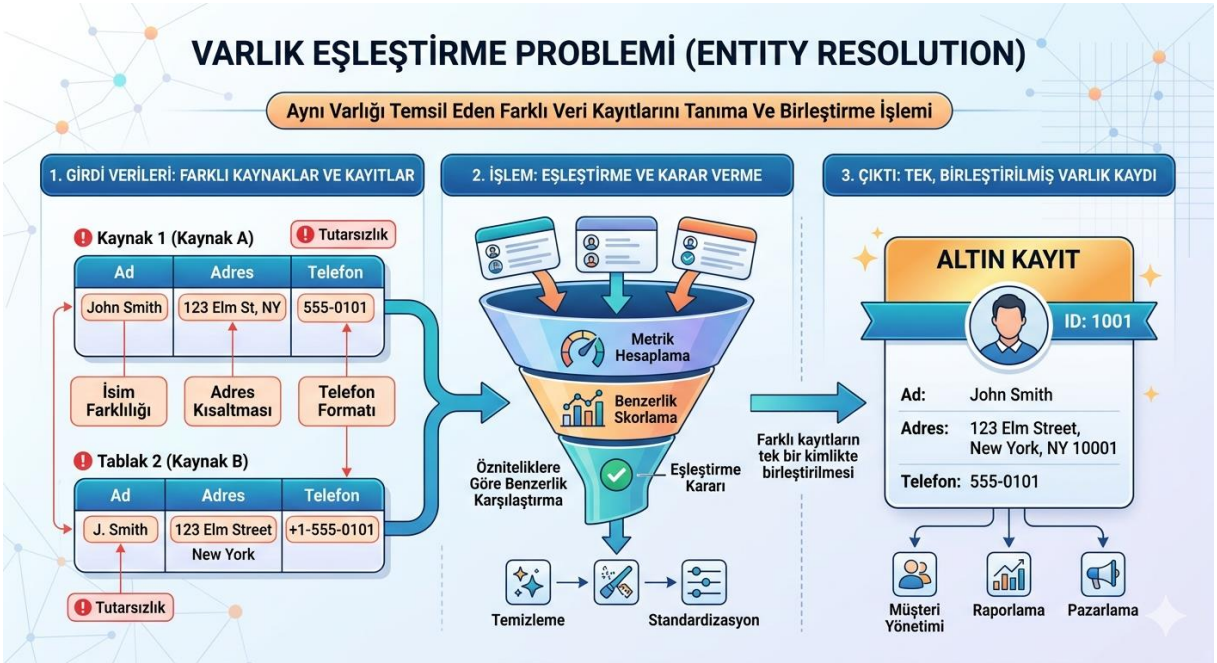
Farklı veri kaynakları aynı varlığı farklı biçimlerde temsil edebilir. Örneğin bir kaynakta “Prof. Dr. Ahmet Yılmaz” olarak geçen bir kişi, başka bir kaynakta “A. Yılmaz” veya yalnızca “Ahmet Yılmaz” şeklinde yer alabilir. Hatta bazı durumlarda yazım hataları, eksik bilgiler veya farklı dillerdeki karşılıklar nedeniyle aynı varlık tamamen farklıymış gibi görünebilir. Bu durum, veri bütünlüğü açısından ciddi bir belirsizlik yaratır.

Varlık eşleştirme problemi yalnızca metin benzerliği üzerinden çözülemez. Çünkü iki varlık ismi birbirine çok benzese bile farklı kişileri temsil edebilir. Örneğin aynı isimde iki farklı kişi farklı şehirlerde veya farklı kurumlarda bulunabilir. Bu nedenle çözüm, yalnızca yüzeysel metin analizine değil, bağlamsal ve ilişkisel analize dayanmalıdır. İşte bu noktada bilgi grafi yaklaşımı devreye girer.

Bilgi grafi sistemlerinde varlık eşleştirme, yalnızca isim benzerliğine değil, ilişkiler ağına bakılarak yapılır. Bir varlığın kimlerle bağlantılı olduğu, hangi olaylara katıldığı ve hangi özellikleri taşıdığı birlikte değerlendirilir. Bu sayede daha güvenilir bir eşleştirme yapılabilir. Örneğin aynı isme sahip iki kişinin farklı kurumlarla ilişkili olması, onların farklı varlıklar olduğunu gösterebilir.

Modern sistemlerde varlık eşleştirme problemi genellikle makine öğrenmesi ve olasılıksal modellerle çözülür. Özellikle gömme (embedding) tabanlı yöntemler, varlıkları vektör uzayında temsil ederek benzerlik hesaplamalarını mümkün kılar. Ancak bu yöntemler tek başına yeterli değildir; graf tabanlı doğrulama mekanizmaları ile desteklenmeleri gerekir.

Şekil 13. Varlık Eşleştirme Problemi



Şekil 13, veri kalitesi, veri ambar yönetimi ve bilgi graflarının inşasında en kritik ön hazırlık adımlarından biri olan Varlık Eşleştirme Problemini (Entity Resolution / Record Linkage) ele alan bir akış şemasıdır. Görsel, en üstteki ana başlığın hemen altında bu işlemi *"Aynı varlığı temsil eden farklı veri kayıtlarını tanıma ve birleştirme işlemi"* olarak çok net bir şekilde tanımlar ve süreci soldan sağa üç ana aşamada modeller:

Sol panelde, sistemlerin gerçek hayatta karşılaştığı en büyük sorun olan "kirli ve tutarsız veri" problemi somut bir örnek üzerinden gösterilmektedir. Burada aynı kişiye ait iki farklı kayıt (Kaynak A ve Kaynak B) yer alır ve aralarındaki tutarsızlıklar kırmızı oklarla işaret edilmiştir:

- **İsim Farklılığı:** Kaynak A'da isim açıkça John Smith yazılmışken, Kaynak B'de J. Smith şeklinde kısaltılmıştır.
- **Adres Kısaltması:** Kaynak A'da adres 123 Elm St, NY şeklinde kısaltmalar barındırırken, Kaynak B'de 123 Elm Street, New York olarak daha uzun ve farklı yazılmıştır.
- **Telefon Formatı:** Kaynak A'da 555-0101 olan numara, Kaynak B'de ülke kodu eklenerek +1-555-0101 şeklinde formatlanmıştır.

Merkezî panel, bu farklı ve dağınık girdilerin bir huni (filtreleme/işleme) mekanizmasından geçirilerek nasıl anlamlandırıldığını gösterir. Süreç adım adım şu algoritmik süzgeçlerden geçer:

- **Özniteliklere Göre Benzerlik Karşılaştırması:** Veriler huniye girdiğinde ilk olarak **Metrik Hesaplama** (örneğin metin benzerlikleri için Jaro-Winkler veya Levenshtein distance gibi metrikler) aşamasına tabi tutulur.
- **Benzerlik Skorlama:** Elde edilen metrikler doğrultusunda kayıtların birbirine ne kadar benzediği puanlanır.
- **Eşleştirme Kararı:** Belirlenen eşik değerinin (threshold) üzerinde kalan kayıtlar için "Evet, bu iki kayıt aynı kişiye aittir" kararı verilir.
- Huninin altında bu sürecin sağlıklı yürüyebilmesi için arka planda çalışan **Temizleme**, **Standardizasyon** ve **Normalizasyon** gibi veri ön işleme adımları mikro ikonlarla sembolize edilmiştir.

Sağ panel, karmaşık sürecin nihai ve en temiz ürününü, yani veri dünyasında **"Altın Kayıt" (Golden Record / Single Source of Truth)** olarak adlandırılan yapıyı sunar.

- **ID: 1001:** Sistem artık iki farklı kayıt yerine, tek bir benzersiz kimlik (ID) üretmiştir.
- **Konsolide Veri:** İsim John Smith olarak netleştirilmiş, adres en açık haliyle (123 Elm Street, New York, NY 10001) birleştirilmiş ve telefon formatı standardize edilmiştir.
- **Kullanım Alanları:** Bu temiz ve tekil "Altın Kayıt", en altta gösterildiği üzere organizasyon içindeki farklı departmanlar ve süreçler için güvenilir bir yakıt haline gelir:
    - **Müşteri Yönetimi (CRM):** Müşteriye mükerrer ulaşımı engeller.

- **Raporlama (BI):** Analitik raporların doğru veriyle çalışmasını sağlar.
- **Pazarlama (Marketing):** Doğru hedefleme yapılmasına olanak tanır.

Bu görsel; veri tabanlarındaki gürültüyü, yazım farklılıklarını ve format hatalarını (Girdi), akıllı eşleştirme ve skorlama algoritmalarıyla (İşlem) eleyerek; sistem genelinde güvenle kullanılabilecek, tekil ve tertemiz bir kurumsal bilgi varlığına (Altın Kayıt) dönüştürme sürecinin tam bir uçtan uca mimari haritasıdır.

Varlık eşleştirme problemi aynı zamanda veri kalitesi ile doğrudan ilişkilidir. Gürültülü, eksik veya tutarsız veri, eşleştirme sürecini zorlaştırır. Bu nedenle veri temizleme süreçleri, bu problemin ayrılmaz bir parçasıdır. Ayrıca büyük ölçekli sistemlerde performans da önemli bir faktördür; milyonlarca varlık arasında doğru eşleştirme yapmak ciddi hesaplama gücü gerektirir.

Sonuç olarak varlık eşleştirme problemi, bilgi grafı sistemlerinin temel yapı taşlarından biridir. Çünkü yanlış eşleştirilmiş bir varlık, tüm grafın anlamsal doğruluğunu bozabilir. Bu nedenle bu problem, yalnızca teknik bir veri problemi değil, aynı zamanda bilginin doğruluğunu belirleyen kritik bir mühendislik ve modelleme problemidir.

3.5 Veri Bütünleştirme

Veri bütünleştirme, farklı kaynaklardan gelen verilerin tek bir tutarlı ve anlamlı sistem içinde birleştirilmesi sürecidir. Bilgi grafı mimarisinde bu süreç, sistemin en temel yapı taşlarından biri olarak kabul edilir. Çünkü gerçek dünya verisi nadiren tek bir kaynaktan gelir; aksine çok sayıda heterojen, dağınık ve farklı formatlara sahip veri kaynağından beslenir.

Şekil 14. Veri Bütünleştirme Süreci

Şekil 14, modern veri mühendisliği ve bilgi mimarisinin en temel süreçlerinden biri olan "Veri Bütünleştirme Süreci" (Data Integration) akışını baştan sona özetleyen kapsamlı bir infografiktir. Şema, soldan sağa doğru ilerleyen 3 ana panelden ve bu panelleri birbirine bağlayan girdi-çıktı oklarından oluşur. Temel amaç; farklı yapılardaki dağınık verileri alıp, temizleyip, işleyerek tutarlı, entegre ve anlamlı bir ortak modele dönüştürmektir.

Bölüm bölüm inceleyecek olursak:

Sürecin başlangıç noktasıdır. Burada verinin tek bir çeşitten değil, çok farklı formatlarda ve yapılarda (heterojen) geldiği bir petek diyagramıyla gösterilmiştir:

- **İlişkisel Veri Tabanları:** SQL tabanlı, satır ve sütunlardan oluşan yapılandırılmış (structured) veriler.
- **API Servisleri:** JSON ve XML formatında gelen, web servisleri aracılığıyla akan yarı yapılandırılmış veriler.
- **Metin ve PDF Belgeleri:** Dokümanlar, raporlar gibi yapılandırılmamış (unstructured) metin verileri.
- **Web Sayfaları:** HTML tabanlı internet verileri.
- **Sensör Verileri:** Nesnelerin İnterneti (IoT) cihazlarından veya cihaz sinyallerinden gelen zaman serisi akışları.
- **Sosyal Medya Akışları:** Kullanıcı etkileşimleri, beğeniler ve paylaşımlardan oluşan anlık veriler.

Kırmızı Ok (Girdi): Bu paneli ortadaki sürece bağlayan ok, girdinin niteliğini açıklar: *"Farklı yapılar, formatlar ve anlam modelleri."* Yani veri henüz ham, karmaşık ve birbiriyle konuşamaz durumdadır.

Ham verinin işlendiği, arındırıldığı ve hizalandığı "mutfak" kısmıdır. Burada farklılıkların ortadan kaldırılması ve standardizasyon hedeflenir:

- **Temizleme (Cleaning):** Süpürge ve kova ikonuyla sembolize edilen bu adımda; eksik, hatalı, yinelenen veya gürültülü veriler ayıklanır.
- **Şema Eşleştirme (Schema Matching):** Farklı kaynaklardan gelen tabloların veya veri yapılarının birbiriyle uyumlu hale getirilmesi sürecidir.
- **Şema Dönüşümü (Schema Transformation):** Verinin hedef formata uygun olarak yapısal olarak değiştirilmesi (örneğin iki farklı tablonun birleştirilmesi).
- **Dönüşüm (Transformation):** Veri tiplerinin dönüştürülmesi (örneğin yazı formatındaki bir tarihin standart tarih formatına getirilmesi).
- **Çözümleme (Parsing/Analytics):** Verinin bileşenlerine ayrılarak anlamlandırılması.

Yeşil Ok (Çıktı): Sürecin başarıyla tamamlandığını gösteren bu ok, *"Ortak temsilin oluşturulması"* aşamasına geçişi sağlar.

Sürecin nihai çıktısıdır. Artık elinizde dağınık dosyalar değil, birbiriyle ilişkili bir **"Bilgi Grafı" (Knowledge Graph) veya Ontoloji** yapısı vardır. Merkezdeki akıllı şehir/bina görselinin etrafındaki ağ yapısı bunu sembolize eder:

- **Kentsel Veri Modeli:** Şehir yapıları (PDF'ler, haritalar), vatandaş profilleri, coğrafi yapılar ve trafik verileri artık birbirinden bağımsız değildir. Hepsi "Özellikler" (Properties) ve "Kavramsal Örnekler" (Concept Examples) bağlamıyla birbirine düğümlerle (nodes) ve ilişkilerle (edges) bağlanmıştır.
- **Alt Bölüm (Veri Sunumu):** En altta ise bu entegre edilmiş yapının teknik olarak nasıl servis edilebileceği gösterilir: Standart bir ilişkisel tablo, temiz bir JSON çıktısı ve dokümante edilmiş bir metin şeması (Text Schema).

Bu şema, "Veri Ambarı" (Data Warehouse), "GraphRAG" veya "Bilgi Grafiği" mimarisi kurarken izlenen mantıksal yol haritasıdır. Dağınıklıktan (sol), standardizasyona (orta) ve nihayetinde anlamlı bir bütüne (sağ) ulaşan veri yaşam döngüsünü görselleştirir.

Günümüzde veri; ilişkisel veri tabanları, API servisleri, metin dosyaları, PDF belgeleri, web sayfaları, sensör verileri ve sosyal medya akışları gibi çok çeşitli kaynaklarda bulunur. Bu kaynakların her biri farklı bir yapı, farklı bir format ve farklı bir anlam modeli içerir. Veri bütünleştirme süreci, bu farklılıkları ortadan kaldırarak ortak bir temsil oluşturmayı amaçlar.

Bilgi grafı sistemlerinde veri bütünleştirme, yalnızca teknik bir birleştirme işlemi değildir; aynı zamanda anlamsal bir uyumlama sürecidir. Çünkü aynı kavram farklı kaynaklarda farklı isimlerle veya farklı yapılarla temsil edilebilir. Örneğin bir veri setinde “müşteri”, başka bir veri setinde “kullanıcı” olarak geçebilir. Bu tür durumlarda anlamsal eşleştirme yapılması gerekir.

Veri bütünleştirmenin en önemli aşamalarından biri şema uyumlamadır (schema alignment). Farklı veri kaynaklarının şemaları birbirine uyarlanarak ortak bir yapı oluşturulur. Bu süreçte alan eşleştirme, veri tiplerinin dönüştürülmesi ve eksik bilgilerin tamamlanması gibi işlemler gerçekleştirilir.

Bir diğer önemli konu veri çakışmalarıdır. Aynı varlık hakkında farklı kaynaklarda çelişkili bilgiler bulunabilir. Örneğin bir kaynakta bir kişinin doğum tarihi farklı, başka bir kaynakta farklı olabilir. Bu durumda hangi bilginin doğru olduğuna karar vermek için güven skoru, kaynak güvenilirliği ve bağlamsal analiz gibi yöntemler kullanılır.

Bilgi grafı yaklaşımı, veri bütünleştirmeyi yalnızca bir ETL (Extract–Transform–Load) süreci olarak değil, sürekli çalışan bir entegrasyon mekanizması olarak görür. Yani veri sadece bir kez birleştirilmez; sistem sürekli olarak yeni verilerle güncellenir ve yeniden organize edilir.

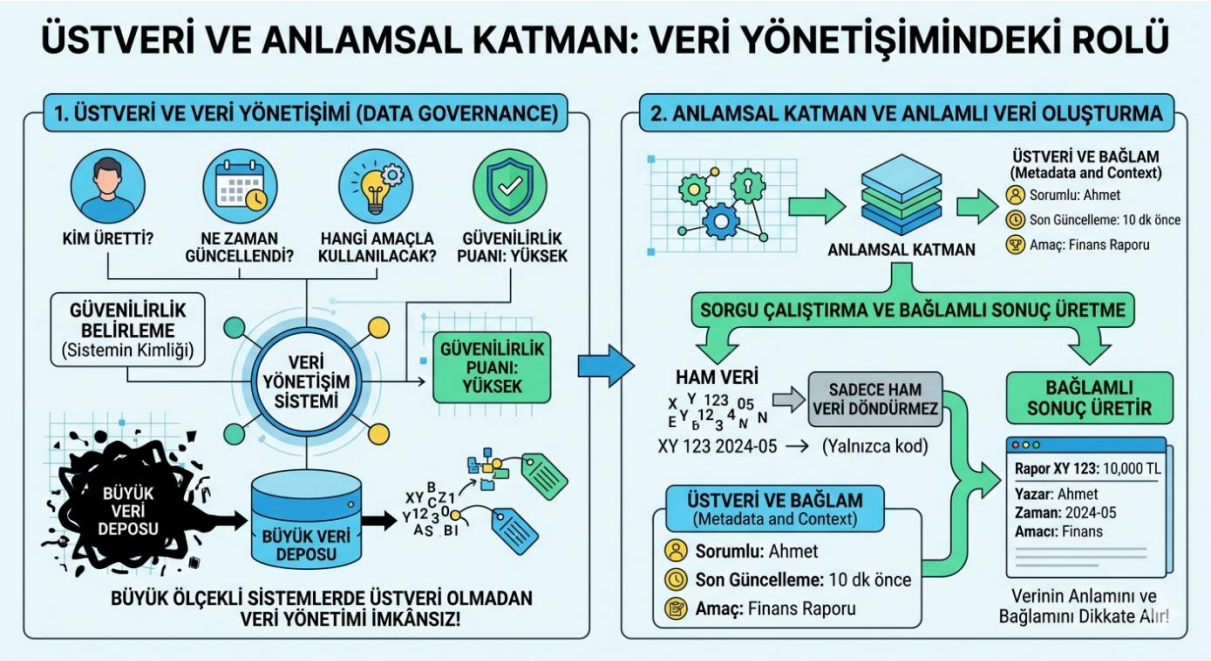
Modern yapay zekâ sistemlerinde veri bütünleştirme, RAG (Retrieval-Augmented Generation) gibi mimarilerin de temelini oluşturur. Çünkü doğru bilgiye erişim, ancak doğru şekilde

bütünleştirilmiş veri ile mümkündür. Sonuç olarak veri bütünleştirme, bilgi grafı mimarisinin “birleştirici dokusu”dur. Farklı kaynakları tek bir anlamlı yapı altında toplar ve böylece dağınık veriyi tutarlı bir bilgi sistemine dönüştürür.

3.6 Üstveri ve Anlamsal Katman

Üstveri (metadata), en basit tanımıyla “veri hakkında veri” anlamına gelir. Ancak bilgi grafı mimarisi içinde üstveri, çok daha derin bir rol oynar. Üstveri, bir verinin ne anlama geldiğini, nereden geldiğini, nasıl üretildiğini ve hangi bağlamda kullanılması gerektiğini tanımlar. Bu nedenle üstveri, yalnızca teknik bir açıklama değil, aynı zamanda anlamsal bir rehberdir.

Şekil 15. Üstveri ve Anlamsal Katman



Şekil 15 bize, büyük veri sistemlerinde verinin başıboş bir yığın olmaktan çıkarılıp nasıl güvenilir ve anlamlı bir kurumsal varlığa dönüştürüldüğünü iki ana aşamada özetlemektedir. Sol tarafta yer alan "Üstveri ve Veri Yönetişimi" bölümü, büyük veri depolarındaki karmaşayı önlemek için verinin "kimliği" niteliğindeki bilgileri ele alır. Bir verinin kim tarafından üretildiği, ne zaman güncellendiği ve hangi amaca hizmet ettiği gibi metaveriler filtrelenerek sisteme bir "Güvenilirlik Puanı" atanır. Görselde de vurgulandığı gibi, bu üstveri mimarisi olmadan büyük ölçekli sistemlerde veriyi yönetmek imkansızdır; çünkü üstveri, verinin izlenebilirliğini ve doğruluğunu garanti altına alan bir denetim mekanizması işlevi görür.

Sağ tarafta bulunan "Anlamsal Katman ve Anlamlı Veri Oluşturma" bölümü ise bu ham ve yönetilen verinin kullanıcı için nasıl işlenebilir hale geldiğini açıklar. Sisteme düz bir sorgu gönderildiğinde, anlamsal katman devreye girmeden sadece "XY 123 2024-05" gibi anlamsız, kuru bir ham veri kodu döner. Ancak üstveri ve bağlamı dikkate alan anlamsal katman sayesinde, sistem bu ham kodu yorumlar ve kullanıcıya "Rapor XY 123: 10.000 TL, Yazar: Ahmet, Zaman: 2024-05, Amacı: Finans" şeklinde insan diline uygun, bağlamı olan anlamlı bir

sonuç üretir. Özetle görsel; üstverinin veriye *güvenilirlik ve düzen*, anlamsal katmanın ise *bağlam ve mantık* kazandırdığını basit bir akışla ortaya koymaktadır.

Anlamsal katman ise üstverinin sistem içinde aktif olarak kullanıldığı katmandır. Bu katman, ham veriyi alır ve onu anlamlı sorgulara, ilişkisel yapılara ve çıkarımsal sonuçlara dönüştürür. Başka bir ifadeyle, anlamsal katman veri ile anlam arasındaki köprüdür.

Geleneksel veri sistemlerinde veri genellikle yalnızca depolanır ve sorgulanır. Ancak bilgi grafi sistemlerinde veri, bağlamı ile birlikte değerlendirilir. Örneğin bir "tarih" alanı yalnızca bir sayı değildir; aynı zamanda bu verinin hangi sistemden geldiği, nasıl üretildiği ve hangi güven seviyesine sahip olduğu da önemlidir. İşte bu bilgiler üstveri tarafından sağlanır.

Üstveri, veri yönetişimi açısından da kritik bir rol oynar. Hangi verinin kim tarafından üretildiği, ne zaman güncellendiği ve hangi amaçla kullanılabileceği gibi bilgiler, sistemin güvenilirliğini belirler. Özellikle büyük ölçekli kurumsal sistemlerde üstveri olmadan veri yönetimi neredeyse imkânsız hale gelir.

Anlamsal katman ise bu üstveriyi kullanarak veriyi anlamlı hale getirir. Örneğin bir sorgu çalıştırıldığında sistem yalnızca ham veriyi döndürmez; aynı zamanda bu verinin bağlamını da dikkate alarak sonuç üretir. Bu, klasik SQL tabanlı sistemlerden önemli bir farktır.

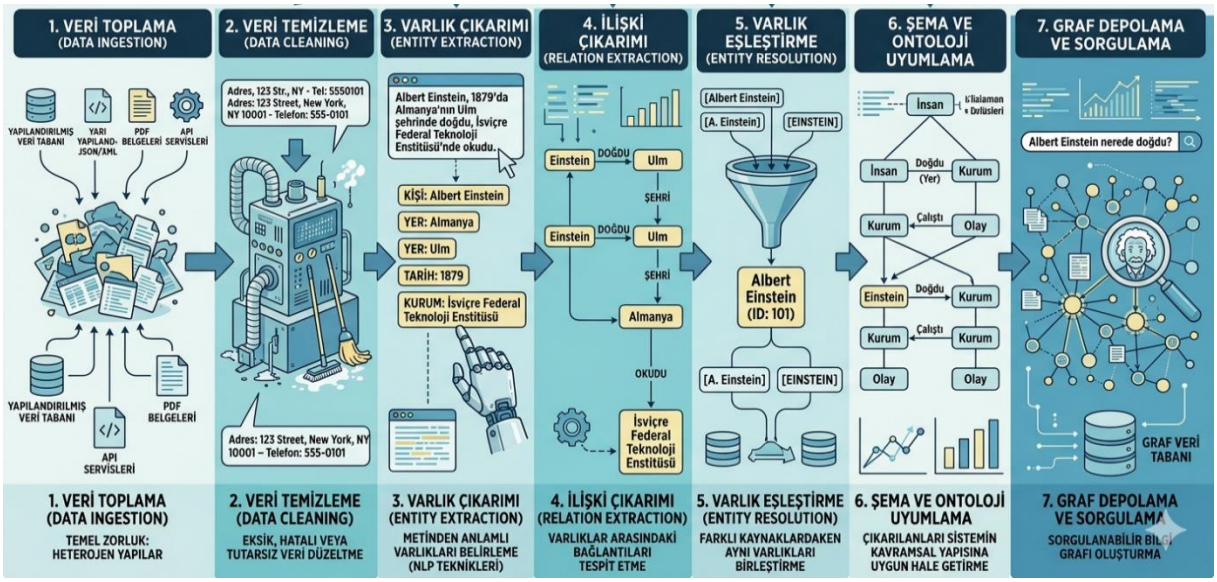
Bilgi grafi sistemlerinde anlamsal katman, ontoloji ile yakından ilişkilidir. Ontoloji, kavramların nasıl tanımlandığını belirlerken, anlamsal katman bu tanımları aktif olarak kullanır. Böylece sistem yalnızca veri depolamaz, aynı zamanda anlam üretir.

Modern yapay zekâ sistemlerinde üstveri ve anlamsal katman, özellikle açıklanabilir yapay zekâ (explainable AI) açısından büyük önem taşır. Çünkü bir sistemin neden belirli bir sonuca ulaştığını anlamak, yalnızca veriyi değil, verinin bağlamını da bilmeyi gerektirir. Sonuç olarak üstveri ve anlamsal katman, bilgi grafi mimarisinin "anlam altyapısını" oluşturur. Veri tek başına ham bir malzeme iken, üstveri ve anlamsal katman sayesinde bu veri yorumlanabilir, sorgulanabilir ve anlamlı bir bilgi sistemine dönüşür.

3.7 Bilgi Grafı İşlem Hattı

Bilgi grafi işlem hattı, ham verinin alınmasından başlayarak anlamlı, sorgulanabilir ve ilişkisel bir bilgi grafına dönüştürülmesine kadar uzanan tüm süreci ifade eder. Bu süreç tek bir adım değil, birbirine bağlı bir dizi aşamadan oluşur ve her aşama, grafın doğruluğunu, tutarlılığını ve kullanım değerini doğrudan etkiler. Bu nedenle bilgi grafi sistemleri, yalnızca veri depolayan yapılar değil, aynı zamanda sürekli çalışan veri işleme hatlarıdır.

Şekil 16. Bilgi Grafı İşlem Hattı



Şekil 16'da görüldüğü gibi işlem hattının ilk aşaması veri toplama sürecidir. Bu aşamada veriler farklı kaynaklardan sisteme çekilir. Bu kaynaklar yapılandırılmış veri tabanları, yarı yapılandırılmış JSON dosyaları, web sayfaları, PDF belgeleri veya API servisleri olabilir. Bu aşamada temel zorluk, verinin heterojen yapısıdır. Her kaynak farklı bir format ve farklı bir anlam düzeni içerir.

İkinci aşama veri temizleme aşamasıdır. Burada eksik, hatalı veya tutarsız veriler düzeltilir. Yazım hataları, format farklılıkları ve tekrar eden kayıtlar bu aşamada ele alınır. Temizleme işlemi yapılmadan bilgi grafına aktarılan veriler, ileride ciddi tutarsızlıklara yol açabilir.

Üçüncü aşama varlık çıkarımı aşamasıdır. Bu aşamada metin veya yapılandırılmamış veriden anlamlı varlıklar belirlenir. Kişiler, kurumlar, yerler, tarihler ve diğer önemli nesneler sistem tarafından tespit edilir. Bu süreç genellikle doğal dil işleme (NLP) teknikleri ile gerçekleştirilir.

Dördüncü aşama ilişki çıkarımı aşamasıdır. Bu aşamada varlıklar arasındaki bağlantılar tespit edilir. Örneğin bir kişinin bir kurumda çalıştığı veya bir makalenin başka bir makaleyi atıf yaptığı gibi ilişkiler ortaya çıkarılır. Bu ilişkiler, bilgi grafının bağlantılarını oluşturur.

Beşinci aşama varlık eşleştirme aşamasıdır. Farklı kaynaklarda geçen aynı varlıkların birleştirilmesi sağlanır. Bu süreç, önceki bölümlerde detaylandırıldığı gibi, bilgi grafının doğruluğu açısından kritik öneme sahiptir.

Altıncı aşama şema ve ontoloji uyumlama aşamasıdır. Çıkarılan varlıklar ve ilişkiler, sistemin önceden tanımlanmış kavramsal yapısına uygun hale getirilir. Bu aşama, verinin anlamsal olarak doğru yerleştirilmesini sağlar.

Son aşama ise grafın depolanması ve sorgulanabilir hale getirilmesidir. Bu noktada graf veri tabanları devreye girer ve sistem artık kullanıcı sorgularına yanıt verebilecek bir bilgi grafı haline gelir.

Tüm bu süreçler birlikte düşünüldüğünde bilgi grafı işlem hattı, statik bir veri dönüşüm süreci değil, sürekli çalışan dinamik bir mühendislik sistemidir. Yeni veriler geldikçe hat yeniden çalışır, graf güncellenir ve sistem kendini sürekli olarak yeniler. Bu yapı, modern yapay zekâ ve büyük veri sistemlerinin temel omurgasını oluşturur.

3.8 Gerçek Zamanlı Bilgi Grafı Sistemleri

Gerçek zamanlı bilgi grafı sistemleri, verinin üretildiği anda işlenip graf yapısına entegre edildiği sistemlerdir. Geleneksel bilgi grafı sistemlerinde veri genellikle toplu (batch) olarak işlenirken, gerçek zamanlı sistemlerde veri akışı sürekli ve kesintisizdir. Bu durum, sistem mimarisini kökten değiştirir ve çok daha karmaşık bir yapı gerektirir.

Gerçek zamanlı sistemlerin temel amacı, dünyadaki değişiklikleri mümkün olan en düşük gecikme ile bilgi grafına yansıtmaktır. Örneğin bir sosyal medya gönderisi, bir finansal işlem veya bir sensör verisi üretildiği anda graf yapısına eklenmelidir. Bu tür sistemlerde gecikme (latency) kritik bir performans ölçütüdür.

Bu mimarinin en önemli bileşenlerinden biri akış işleme (stream processing) katmanıdır. Bu katman, sürekli akan veriyi alır, filtreler, dönüştürür ve graf yapısına uygun hale getirir. Apache Kafka gibi mesajlaşma sistemleri veya benzeri akış altyapıları bu amaçla kullanılır.

İkinci önemli bileşen gerçek zamanlı varlık güncelleme mekanizmasıdır. Yeni bir veri geldiğinde, sistem yalnızca yeni bir düğüm eklemekle kalmaz, aynı zamanda mevcut ilişkileri de güncelleyebilir. Bu durum, grafın sürekli evrilen bir yapı olmasını sağlar.

Gerçek zamanlı sistemlerde tutarlılık ve hız arasında bir denge kurmak gerekir. Çok sık güncellenen sistemlerde tutarlılığı sağlamak zorlaşabilir. Bu nedenle bazı sistemler “nihai tutarlılık” (eventual consistency) yaklaşımını benimser; yani sistem kısa süreli tutarsızlıkları kabul eder ancak zamanla doğru duruma ulaşır. Bu tür sistemler özellikle finans, siber güvenlik, lojistik ve sosyal medya analitiği gibi alanlarda kritik öneme sahiptir. Örneğin bir dolandırıcılık tespit sistemi, anlık işlem verilerini analiz ederek şüpheli bağlantıları hemen ortaya çıkarabilir.

Bilgi grafı açısından gerçek zamanlı sistemler, statik bilgi modellerinden dinamik bilgi modellerine geçişi temsil eder. Artık bilgi yalnızca depolanan bir şey değil, sürekli akan ve değişen bir yapıdır. Bu durum, graf düşüncesini daha da güçlü hale getirir çünkü dünya da zaten sürekli değişen bir ağdır.

Sonuç olarak gerçek zamanlı bilgi grafı sistemleri, modern veri mühendisliğinin en ileri noktalarından birini temsil eder. Bu sistemler sayesinde bilgi grafı yalnızca geçmişi modelleyen bir yapı olmaktan çıkar, aynı zamanda “şimdi”yi de temsil eden canlı bir organizmaya dönüşür.

3.9 Graf Destekli Sorgu ile Üretim Mimarileri (Graf Retrieval-Augmented Generation Architectures / GrafRAG)

Graf destekli sorgu ile üretim mimarileri, bilgi grafı sistemleri ile büyük dil modellerini (LLM) birleştiren yeni nesil yapay zekâ mimarileridir. Bu yaklaşım, özellikle “GrafRAG” olarak

adlandırılan yapılarla öne çıkmaktadır. Temel fikir, dil modellerinin yalnızca eğitim verisine dayanarak değil, aynı zamanda dış bir bilgi grafından dinamik olarak bilgi çekerek yanıt üretmesidir.

Geleneksel büyük dil modelleri, öğrenilmiş parametreler üzerinden çalışır ve bu nedenle güncel olmayan veya eksik bilgi üretebilir. Buna karşılık GrafRAG mimarileri, bilgi grafını bir dış bellek (external memory) olarak kullanır. Böylece model, soruya yanıt vermeden önce graf üzerinde sorgu yaparak ilgili varlıkları ve ilişkileri çeker.

Bu süreç üç temel aşamadan oluşur. İlk aşama sorgu analizi aşamasıdır. Kullanıcının sorusu analiz edilir ve hangi varlıkların ve ilişkilerin gerekli olduğu belirlenir. İkinci aşama graf sorgulama aşamasıdır. Bu aşamada bilgi grafı üzerinde gezinme işlemleri yapılır ve ilgili alt graf çıkarılır. Üçüncü aşama ise üretim aşamasıdır. Burada dil modeli, graftan gelen bilgiyi kullanarak doğal dilde bir yanıt üretir.

GrafRAG mimarilerinin en önemli avantajlarından biri halüsinasyon problemini azaltmasıdır. Çünkü model yalnızca kendi iç belleğine değil, doğrulanabilir dış bir bilgi kaynağına da erişir. Bu durum, özellikle kritik bilgi gerektiren alanlarda güvenilirliği artırır.

Ayrıca bu mimariler açıklanabilirlik (explainability) açısından da güçlüdür. Çünkü üretilen yanıtın hangi graf düğümlerine ve ilişkilerine dayandığı izlenebilir. Bu sayede kullanıcı, modelin neden belirli bir sonuca ulaştığını anlayabilir.

GrafRAG yaklaşımı, bilgi grafı ve yapay zekâ entegrasyonunun en ileri örneklerinden biridir. Bu yapı sayesinde bilgi, yalnızca statik bir veri tabanında saklanmaz; aynı zamanda üretim sürecinin aktif bir parçası haline gelir.

Sonuç olarak graf destekli üretim mimarileri, yapay zekânın geleceğinde önemli bir rol oynamaktadır. Bu sistemler, bilgi grafını yalnızca bir veri yapısı olmaktan çıkararak, akıllı sistemlerin düşünme sürecinin merkezine yerleştirir.

BÖLÜM 4 — GRAF VERİ TABANLARI VE NEO4J

Graf veri tabanları, veriyi yalnızca saklamaya değil, veriler arasındaki ilişkileri anlamaya odaklanan yeni nesil veri sistemleridir. Klasik ilişkisel veri tabanlarında bilgi satırlar, sütunlar ve tablolar halinde tutulurken; graf veri tabanlarında temel yapı taşları düğümler ve bu düğümler arasındaki ilişkilerdir. Bu yaklaşım özellikle karmaşık bağlantıların yoğun olduğu sistemlerde büyük avantaj sağlar. Sosyal ağlar, tedarik zincirleri, finansal işlem ağları, öneri sistemleri, biyolojik ağlar ve bilgi grafları gibi alanlar doğal olarak graf yapısındadır. Bu nedenle graf veri tabanları, gerçek dünyanın ilişkisel doğasına klasik tablolardan çok daha yakın bir model sunar.

Bu bölümde, graf düşüncesinin temel mantığını ve bu yaklaşımın neden giderek daha önemli hale geldiğini inceleyeceğiz. İlk olarak ilişkisel veri tabanlarının özellikle çok katmanlı bağlantılar karşısında yaşadığı sınırlara değinilecek; ardından özellik graf modeli ve RDF yaklaşımı gibi temel graf veri modelleri ele alınacaktır. Daha sonra günümüzde en yaygın graf veri tabanlarından biri olan Neo4j üzerinden graf tabanlı bu mimarinin nasıl çalıştığı açıklanacaktır.

Bölüm boyunca Cypher sorgu dili kullanılarak graf üzerinde nasıl düşünüldüğü gösterilecek; SQL'in JOIN merkezli yaklaşımı ile Neo4j'nin ağ üzerinde gezinme temelli yaklaşımı karşılaştırılacaktır. Özellikle tedarik zinciri örneği üzerinden ürünler, bileşenler, fabrikalar ve lojistik ağları arasında ilişkilerin nasıl modellendiği ve analiz edildiği adım adım ele alınacaktır.

Bu bölümün temel amacı yalnızca yeni bir veri tabanı teknolojisini öğretmek değildir. Asıl amaç, veriyi bağımsız kayıtlar kümesi olarak değil, yaşayan bir ilişkiler ağı olarak düşünmeyi öğretmektir. Çünkü modern veri çağında anlam, çoğu zaman tek tek verilerde değil; veriler arasındaki bağlantıların içinde saklıdır.

4.1 Neden Yeni Bir Veri Modeline Gerek Duyuldu?

Veri tabanlarının evrimi, aslında bilgisayar biliminin gerçek dünyayı temsil etme biçiminin evrimidir. İlk dönemlerde veri, doğrudan hesaplama ve kayıt tutma amacıyla tasarlanmıştı. Bu dönemin en büyük başarısı ilişkisel veri tabanı modelidir. İlişkisel model, Edgar F. Codd tarafından ortaya konduğunda devrim niteliğindeydi; çünkü veriyi matematiksel bir temele, yani kümeler teorisine dayandırıyordu. Bu sayede veri, tablolar içinde düzenli, tutarlı ve sorgulanabilir bir yapıya kavuştu.

İlişkisel veri tabanlarının tarihsel başarısı yadsınamaz. Bankacılık sistemleri, kurumsal ERP yazılımları, muhasebe sistemleri ve neredeyse tüm klasik iş uygulamaları bu model üzerine inşa edildi. SQL dili, bu yapının standart sorgulama aracı haline geldi ve yıllar içinde endüstriyel bir standarda dönüştü. Özellikle veri bütünlüğü, işlem yönetimi ve ACID prensipleri[9] sayesinde

[9] Veri tabanı yönetim sistemlerinde bir işlemin güvenilir bir şekilde gerçekleşmesini garanti altına alan dört temel ilkeye ACID prensipleri (Atomicity, Consistency, Isolation, Durability) denir.

ilişkisel veri tabanları güvenilirliğin sembolü haline geldi. Ancak bu başarı, modelin sınırlarını da görünmez hale getirdi. Çünkü ilişkisel model, veriyi çok güçlü şekilde "tablo içinde" düşünürken, "ilişkiyi" ikinci sınıf bir kavram olarak ele alıyordu.

Zamanla veri hacmi arttı, ancak daha önemlisi veri yapıları karmaşıklaştı. Sosyal ağlar, e-ticaret platformları, lojistik zincirleri, biyolojik sistemler ve finansal ağlar ortaya çıktıkça, verinin doğasının aslında düz tablolardan çok uzak olduğu daha net görülmeye başlandı. Gerçek dünya, satır ve sütunlardan oluşan düzenli bir yapı değil; sürekli bağlantılarla birbirine bağlanan dinamik bir ağdı.

Bu noktada ilişkisel veri tabanlarının en temel problemi ortaya çıktı: JOIN patlaması. Basit görünen bir soruyu yanıtlamak için bile çok sayıda tabloyu birbirine bağlamak gerekiyordu. Örneğin bir kullanıcı, bir ürün, o ürünün tedarikçisi, tedarikçinin fabrikası ve fabrikanın dağıtım ağı gibi çok katmanlı ilişkiler söz konusu olduğunda SQL sorguları giderek daha karmaşık hale geliyordu. Her JOIN işlemi, aslında veri modeline yeni bir zihinsel katman eklemek anlamına geliyordu. Bu durum yalnızca sorguları zorlaştırmakla kalmıyor, aynı zamanda performans açısından da ciddi maliyetler doğuruyordu.

Çok ilişkili veride performans sorunları özellikle büyük ölçekli sistemlerde belirgin hale geldi. Bir sorgu, birkaç tabloyu JOIN ettiğinde kabul edilebilir hızda çalışırken, beş, altı veya daha fazla JOIN içeren sorgularda performans dramatik şekilde düşüyordu. Bunun nedeni, ilişkisel modelin veriyi fiziksel olarak ilişkiler üzerinden değil, dış anahtarlar (foreign keys) ve sonradan hesaplanan bağlantılar üzerinden kurmasıydı. Her JOIN işlemi, aslında veri üzerinde yeniden bir ilişki kurma operasyonuydu. Bu da sistem büyüdükçe hesaplama maliyetinin katlanarak artmasına neden oluyordu.

Bu sorun yalnızca performansla sınırlı değildi; aynı zamanda modelleme problemi de yaratıyordu. Gerçek dünyadaki ağ yapıları, tablo modeline zorla dönüştürüldüğünde anlam kaybı yaşanıyordu. Örneğin bir tedarik zinciri doğal olarak bir graf yapısıdır: ürünler bileşenlerden oluşur, bileşenler fabrikalarda üretilir, fabrikalar şehirlerle bağlantılıdır ve tüm bu ilişkiler dinamik olarak değişir. Ancak bu yapı ilişkisel modele aktarıldığında, her ilişki ayrı bir tabloya bölünmek zorunda kalır. Sonuç olarak doğal bir ağ yapısı, parçalanmış ve dolaylı bir temsil haline gelir.

Bu durumun en önemli sonucu, veri modelinin artık gerçekliği doğrudan temsil edememesi oldu. Model ile dünya arasındaki mesafe arttıkça, sistemler daha karmaşık hale geldi. Geliştiriciler, aslında var olmayan yapay ara tablolarla gerçek ilişkileri yeniden inşa etmek zorunda kaldılar. Bu da hem zihinsel yükü hem de teknik karmaşıklığı artırdı.

Tüm bu sınırlamalar, yeni bir veri modeline olan ihtiyacı doğurdu. Bu yeni modelin temel varsayımı şuydu: Gerçek dünya aslında bir tablo değil, bir ağdır. İnsanlar, nesneler, olaylar ve süreçler birbirine bağlantılarla bağlıdır ve bu bağlantılar, verinin kendisi kadar önemlidir. Bu bakış açısı, veri merkezli düşünceden ilişki merkezli düşünceye geçişi temsil eder.

Veri merkezli düşüncede odak noktası tekil kayıtlar, yani "nesneler"dir. Örneğin bir müşteri, bir ürün ya da bir sipariş bağımsız varlıklar olarak ele alınır. Ancak ilişki merkezli düşüncede odak noktası bu varlıkların nasıl bağlandığıdır. Bir müşteri yalnızca bir kayıt değildir; satın alma

davranışları, sosyal bağlantıları, ürün tercihleri ve zaman içindeki etkileşimleriyle birlikte anlam kazanır. Aynı şekilde bir ürün de yalnızca bir envanter öğesi değil, tedarik zinciri içindeki konumu, üretim bağımlılıkları ve dağıtım yolları ile birlikte değerlendirilir.

Bu dönüşüm, yalnızca teknik bir değişim değildir; aynı zamanda epistemolojik bir değişimdir. Artık veri, "ne var?" sorusuna cevap vermekten çok "neyle bağlı?" sorusuna cevap vermek zorundadır. Çünkü modern sistemlerde değer, çoğu zaman varlıklarda değil, varlıklar arasındaki ilişkilerin yapısında gizlidir.

Sonuç olarak, yeni veri modellerine duyulan ihtiyaç üç temel gerçeğe dayanır: Gerçek dünya ağ yapısındadır, ilişkisel modeller bu ağı doğal biçimde temsil edemez ve ölçek büyüdükçe JOIN tabanlı yaklaşım sürdürülemez hale gelir. Bu nedenle graf veri tabanları, yalnızca alternatif bir teknoloji değil, aynı zamanda veriyi anlama biçiminde köklü bir paradigma değişimi olarak ortaya çıkmıştır.

4.2 Graf Düşüncesi: Veri Değil İlişki Merkezlidir

Graf düşüncesi, veri modellemesinde köklü bir bakış açısı değişimini temsil eder. Klasik veri yaklaşımlarında odak noktası "veri nesneleri"dir; yani sistem önce varlıkları tanımlar, ardından bu varlıklar arasında ilişkiler kurmaya çalışır. Graf yaklaşımında ise bu sıralama tersine döner: önce ilişkiler vardır, sonra varlıklar bu ilişkiler içinde anlam kazanır. Bu nedenle graf modelinin merkezinde veri değil, ilişki bulunur.

Bu yapının temel bileşeni düğüm kavramıdır. Düğüm, gerçek dünyadaki herhangi bir varlığı temsil eder. Bu bir kişi, bir ürün, bir şehir, bir işlem ya da bir sensör olabilir. Düğümün kendisi tek başına anlamlı olabilir, ancak graf yaklaşımında asıl değer, bu düğümün diğer düğümlerle nasıl bağlandığında ortaya çıkar. Yani bir düğüm, izolasyonda değil, bir ağın parçası olarak anlam taşır.

Bu bağlantıları temsil eden yapı bağlantı veya ilişki olarak adlandırılır. İlişki, iki düğüm arasındaki ilişkiyi tanımlar ve bu ilişki yalnızca bir bağlantı değil, aynı zamanda anlam taşıyan bir veri elemanıdır. Örneğin "satın alır", "üretir", "bağlıdır", "transfer eder" gibi ilişkiler yalnızca iki varlığı bağlamakla kalmaz, aynı zamanda bu bağlantının doğasını da tanımlar. Bu nedenle graf modelinde ilişkiler pasif çizgiler değil, aktif veri bileşenleridir.

Graf modelinin önemli özelliklerinden biri özellik mantığıdır. Hem düğümler hem de ilişkiler kendi özelliklerine sahip olabilir. Örneğin bir "müşteri" düğümü ad, yaş ve segment gibi özellikler taşırken, "satın alır" ilişkisi tarih, miktar ve fiyat gibi bilgiler içerebilir. Bu yaklaşım, veriyi yalnızca varlık düzeyinde değil, ilişki düzeyinde de zenginleştirir. Bu da özellikle analitik sistemlerde çok katmanlı bir anlam üretir.

Bir diğer önemli kavram yönlü ilişkilerdir. Graf modelinde ilişkiler genellikle yönlüdür; yani A düğümü B düğümüne bağlı olabilir ama bu bağlantı ters yönde aynı anlamı taşımayabilir. Örneğin bir müşteri "ürünü satın alır", ancak ürün müşteriyi satın almaz. Bu yönlü yapı, gerçek dünyadaki asimetrik ilişkileri doğal biçimde temsil eder.

Graf yapısının en güçlü yönlerinden biri çok katmanlı bağlantıları temsil edebilmesidir. Gerçek dünyada ilişkiler tek katmanlı değildir. Bir kullanıcı yalnızca bir ürünü satın almaz; aynı zamanda o ürün belirli bir tedarik zinciri üzerinden gelir, belirli bir fabrikada üretilir, belirli bir lojistik ağından geçer ve başka kullanıcıların davranışlarını da etkiler. Bu çok katmanlı yapı, graf modelinde doğal olarak temsil edilebilir. Düğümler ve ilişkiler arasında sınırsız derinlikte bağlantılar kurulabilir.

Bu yaklaşım "önce ilişki modellemesi" olarak adlandırılır. Bu modelleme biçiminde tasarımın başlangıç noktası varlıklar değil, ilişkiler ağının kendisidir. Önce sistemde hangi tür ilişkilerin bulunduğu düşünülür, ardından bu ilişkileri taşıyacak düğümler tanımlanır. Bu, özellikle karmaşık sistemlerin modellenmesinde çok daha doğal ve sezgisel bir yaklaşım sağlar.

Graf düşüncesinin gücü, gerçek dünya problemlerinde açıkça görülür. Sosyal ağlar bunun en klasik örneğidir. İnsanlar düğümleri, arkadaşlık veya takip ilişkileri ise bağlantıları oluşturur. Bu yapı sayesinde etki analizi, öneri sistemleri ve topluluk tespiti gibi problemler doğal olarak çözülebilir.

Finans ağları da graf yapısına mükemmel bir örnektir. Banka hesapları, transfer işlemleri ve kurumlar arasındaki para akışı, yoğun bir ilişki ağı oluşturur. Dolandırıcılık tespiti gibi problemler, bu ilişkiler arasındaki anormal desenlerin bulunmasıyla çözülür.

Tedarik zincirleri ise graf modelinin en somut endüstriyel kullanım alanlarından biridir. Ürünler, bileşenler, fabrikalar ve dağıtım noktaları birbiriyle çok katmanlı ilişkiler içindedir. Bu yapı, graf modelinde doğal olarak temsil edilir ve risk analizi, optimizasyon ve bağımlılık analizi gibi işlemler kolaylaşır.

Son olarak bilgi ağları, yani bilgi grafı yapıları, graf düşüncesinin en soyut ama en güçlü kullanım alanıdır. Kavramlar, belgeler, kişiler ve olaylar arasındaki semantik ilişkiler, bilginin kendisini bir ağ yapısına dönüştürür. Bu sayede bilgi yalnızca depolanmaz, aynı zamanda bağlam içinde anlam kazanır. Sonuç olarak graf düşüncesi, veriyi tekil nesneler olarak değil, birbirine bağlı ilişkiler bütünü olarak görmeyi sağlar. Bu bakış açısı, modern veri sistemlerinin temel zihinsel dönüşümünü temsil eder: önemli olan şey neyin var olduğu değil, neyle bağlı olduğudur.

4.3 Graf Veri Modelleri

Graf veri tabanları tek bir yaklaşımın ürünü değildir; aksine farklı ihtiyaçlara cevap veren iki ana model etrafında şekillenir: Özellik Grafı (Property Graph) modeli ve RDF (Kaynak Tanımlama Çerçevesi – Resource Description Framework) modeli. Bu iki yaklaşım, aynı temel fikri —veriyi düğümler ve ilişkiler üzerinden temsil etmeyi— paylaşsa da, felsefi ve teknik düzeyde önemli farklılıklara sahiptir. Bu bölümde her iki modelin yapısı, mantığı ve kullanım alanları karşılaştırmalı olarak ele alınacaktır.

Özellik Grafı Modeli

Özellik Grafı modeli, günümüzde endüstride en yaygın kullanılan graf veri modelidir. Bu modelde temel yapı üç bileşenden oluşur: düğüm, ilişki ve özellik. Düğümler gerçek dünyadaki

varlıkları temsil ederken, ilişkiler bu varlıklar arasındaki bağlantıları ifade eder. Özellikler ise hem düğümlere hem de ilişkilere eklenebilen açıklayıcı veri parçalarıdır.

Örneğin bir "Müşteri" düğümü ad, yaş ve segment gibi özelliklere sahip olabilir. Aynı şekilde "SATIN_ALIR" ilişkisi yalnızca iki düğümü bağlamakla kalmaz, aynı zamanda tarih, fiyat ve miktar gibi özellikler de taşıyabilir. Bu özellik, graf modelini yalnızca yapısal değil, aynı zamanda anlamsal açıdan da zengin hale getirir.

Özellik Grafı modelinin önemli bileşenlerinden biri etiket mantığıdır. Etiketler, düğümlerin türünü tanımlamak için kullanılır. Bir düğüm aynı anda birden fazla etikete sahip olabilir. Örneğin bir düğüm hem "Kişi" hem de "Müşteri" etiketini taşıyabilir. Bu esneklik, modelin katı şemalara bağımlı olmadan çalışmasını sağlar.

Bir diğer kritik kavram ilişki özellikleri yapısıdır. Klasik veri modellerinde ilişkiler yalnızca bağlantı görevi görürken, Özellik Grafı modelinde ilişkiler de veri taşır. Bu durum özellikle zaman boyutu içeren ilişkilerde veya ağırlıklı bağlantılarda büyük avantaj sağlar. Örneğin bir lojistik ağında "SEVKİYAT" ilişkisi yalnızca iki konumu bağlamakla kalmaz, aynı zamanda maliyet, süre ve taşıma kapasitesi gibi bilgileri de içerir.

Bu modelin en güçlü yönlerinden biri şema esnekliğidir. Önceden tanımlanmış katı bir tablo yapısı yoktur. Yeni düğüm türleri veya yeni ilişki tipleri sisteme eklenebilir ve mevcut yapı bozulmadan genişletilebilir. Bu özellik, özellikle hızlı değişen veri yapılarında büyük avantaj sağlar. Veri modeli, uygulamanın evrimine paralel olarak büyüyebilir.

RDF ve Üçlü Modeli

RDF modeli (Kaynak Tanımlama Çerçevesi), graf veri yaklaşımının daha semantik ve standartlaştırılmış bir versiyonudur. Bu modelin temel yapı taşı üçlü yapı yaklaşımıdır: Özne – Yüklem – Nesne.

Bu yapıda özne bir varlığı, yüklem bu varlıkla ilgili bir özelliği veya ilişkiyi, nesne ise bu ilişkinin hedefini temsil eder. Örneğin:

"Ali — çalışır — ŞirketX"

burada "Ali" özne, "çalışır" yüklem ve "ŞirketX" nesne olur.

RDF modelinin en önemli özelliklerinden biri URI (Tekdüzen Kaynak Tanımlayıcı – Uniform Resource Identifier) kullanımıdır. Her varlık ve ilişki küresel olarak tanımlanabilir bir kimliğe sahiptir. Bu sayede farklı sistemlerdeki veriler birleştirilebilir ve standart bir anlamsal yapı oluşturulabilir.

RDF yaklaşımı aynı zamanda ontoloji kavramı ile yakından ilişkilidir. Ontoloji, bir alanın kavramsal modelini tanımlar; yani hangi varlık türlerinin bulunduğunu, bu varlıklar arasındaki ilişkileri ve bu ilişkilerin kurallarını belirler. Bu yapı, Anlamsal Web (Semantic Web) vizyonunun temelini oluşturur.

Anlamsal Web yaklaşımı, veriyi yalnızca makinelerin sakladığı bir bilgi değil, aynı zamanda anlamlı ve makine tarafından yorumlanabilir bir yapı haline getirmeyi hedefler. RDF bu vizyonun teknik temelidir. Özellikle bağlantılı veri projeleri, akademik bilgi sistemleri ve açık veri platformlarında yaygın olarak kullanılır.

Özellik Grafı ve RDF

Özellik Grafı ve RDF modelleri aynı problemi çözmeye çalışsa da farklı önceliklere sahiptir. Bu farklar hem yapısal hem de kullanım düzeyinde kendini gösterir. Yapısal farklar açısından Özellik Grafı modeli daha esnek ve uygulama odaklıdır. Düğümler ve ilişkiler doğrudan özellik taşıyabilir. RDF ise daha atomik bir yapı sunar; her şey üçlü formatına indirgenir ve genişletilebilirlik ontoloji üzerinden sağlanır.

Gezinme farkı oldukça kritiktir. Özellik Grafı modelinde gezinme işlemleri doğrudan düğümler arasındaki işaretçi benzeri bağlantılar üzerinden yapılır. Bu, özellikle derin bağlantı analizlerinde yüksek performans sağlar. RDF modelinde ise sorgular genellikle SPARQL dili üzerinden üçlü desen eşleştirme (triple pattern matching) ile gerçekleştirilir; bu daha bildirime dayalıdır ancak bazı durumlarda daha maliyetli olabilir.

Performans farkı bu noktada belirginleşir. Özellik Grafı, özellikle operasyonel sistemlerde ve gerçek zamanlı sorgularda daha hızlıdır. RDF ise büyük ölçekli anlamsal veri entegrasyonu ve bilgi birleştirme senaryolarında daha güçlüdür. Anlamsal fark açısından RDF daha katı ve standartlaştırılmış bir anlam modeli sunar. Ontoloji temelli olduğu için veri paylaşımı ve birlikte çalışabilirlik çok güçlüdür. Özellik Grafı ise daha pragmatiktir; uygulama ihtiyaçlarına göre hızlı modelleme yapılmasına izin verir.

> Peki ne zaman hangisi tercih edilmelidir? Eğer amaç operasyonel bir sistem kurmak, hızlı sorgular yapmak ve dinamik veri üzerinde çalışmak ise Özellik Grafı modeli daha uygundur. Eğer amaç farklı veri kaynaklarını birleştirmek, güçlü bir anlamsal yapı kurmak ve ontoloji tabanlı bilgi sistemleri geliştirmek ise RDF modeli daha doğru bir seçimdir. Sonuç olarak bu iki model birbirine rakip değil, farklı problem sınıflarına yönelik iki farklı düşünme biçimidir. Özellik Grafı “nasıl bağlanır?” sorusuna odaklanırken, RDF “ne anlama gelir?” sorusuna odaklanır. Bu ayrım, graf veri dünyasının hem mühendislik hem de anlamsal boyutunu anlamak için kritik bir eşiktir.

4.4 Neo4j: Graf Tabanlı Veri Tabanı

Neo4j, graf veri tabanları arasında en yaygın kullanılan ve endüstri standardı haline gelmiş sistemlerden biridir. Onu özel kılan şey yalnızca graf veri modelini desteklemesi değil, doğrudan graf düşüncesi üzerine inşa edilmiş bir mimariye sahip olmasıdır. Yani Neo4j, sonradan graf desteği eklenmiş bir sistem değil; en baştan itibaren ilişkileri birinci sınıf veri öğesi olarak kabul eden bir tasarım felsefesiyle geliştirilmiştir.

Neo4j’nin ortaya çıkışı, ilişkisel veri tabanlarının karmaşık ilişki sorgularında zorlanmaya başladığı döneme dayanır. Özellikle çok katmanlı bağlantıların (multi-hop relationships) yoğun

olduğu sistemlerde SQL sorgularının JOIN zincirlerine dönüşmesi, performans ve modelleme açısından ciddi sınırlamalar yaratmıştır. Neo4j bu probleme doğrudan bir alternatif olarak geliştirilmiş ve ilişkileri verinin merkezine yerleştiren bir yaklaşım benimsemiştir.

Graf tabanlı yaklaşımın temel farkı şudur: sistem, veriyi tabloya dönüştürmez, doğrudan graf olarak saklar. Bu yaklaşımın en kritik bileşeni indeks bağımsız komşuluk (index-free adjacency) kavramıdır. **"İndeks bağımsız komşuluk"**, verinin kendisinin ve ilişkilerinin bellekte fiziksel olarak birbirine bağlı olması durumudur. Graf veri tabanlarını (örneğin Neo4j) ilişkisel veri tabanlarından ayıran ve "ilişki odaklı" yapan ana yetenek tam olarak budur. Bu kavrama göre her düğüm, doğrudan komşularına fiziksel referanslar ile bağlıdır. Bu sayede bir düğümden diğerine geçiş için ek bir indeks araması veya JOIN işlemi gerekmez.

Gezinme mantığı tam olarak bu yapının üzerine kuruludur. Gezinme, graf üzerinde düğümden düğüme geçerek ilişkileri takip etme işlemidir. Neo4j'de sorgular, tablo taraması veya JOIN hesaplaması yerine doğrudan bu gezinme mekanizması üzerinden çalışır. Bu durum, özellikle derin bağlantıların analiz edildiği senaryolarda büyük performans avantajı sağlar.

Bu noktada en kritik soru ortaya çıkar: Neden JOIN gerekmez? Çünkü ilişkisel veri tabanlarında iki veri parçası arasındaki bağlantı, çalışma zamanında hesaplanır. Neo4j'de ise bu bağlantı fiziksel olarak zaten mevcuttur. Yani ilişki, sonradan türetilen bir sonuç değil, verinin kendisidir. Bu nedenle sistem, JOIN hesaplaması yapmak yerine doğrudan ilişkiyi takip eder.

İlişkilerin fiziksel olarak saklanması Neo4j'nin temel mimari avantajlarından biridir. Her ilişki, disk üzerinde bağımsız bir veri yapısı olarak tutulur ve hem başlangıç hem de bitiş düğümlerine referans içerir. Bu yapı, grafın yalnızca mantıksal değil, aynı zamanda fiziksel olarak da bir ağ yapısı şeklinde saklanmasını sağlar.

Gezinme performansı bu mimarinin doğal bir sonucudur. Bir düğümden başlayarak çok sayıda ilişkiyi takip etmek, sabit zamanlı veya çok düşük maliyetli işlemlerle gerçekleştirilebilir. Bu durum özellikle büyük ölçekli ağ analizlerinde ve çok adımlı sorgularda (multi-hop queries) ciddi performans avantajı yaratır.

Neo4j'nin en önemli kavramsal metaforlarından biri "yaşayan graf" (living graph) yaklaşımıdır. Bu yaklaşıma göre graf, statik bir veri yapısı değil, sürekli büyüyen ve değişen bir organizmadır. Yeni düğümler ve ilişkiler eklendikçe sistem yeniden yapılandırılmaz; graf doğal olarak genişler. Bu özellik, özellikle dinamik veri ortamlarında büyük esneklik sağlar.

4.4.1 Neo4j'nin İç Mimarisi

Neo4j'nin iç mimarisi, graf modelinin fiziksel dünyadaki karşılığını oluşturur. Bu mimarinin merkezinde veri depolama motoru) bulunur. BU motor düğümleri, ilişkileri ve özellikleri disk üzerinde optimize edilmiş bir yapı halinde saklar.

Bu yapının en kritik bileşenlerinden biri pointer tabanlı ilişki yapısıdır. Her düğüm, kendisine bağlı ilişkilerin adreslerini doğrudan tutar. Bu sayede sistem, bir düğümden diğerine geçerken arama yapmak yerine doğrudan referans üzerinden ilerler. Bu yapı, geleneksel indeks tabanlı sistemlere göre çok daha düşük gecikme sağlar.

Cache mekanizması, Neo4j'nin performansında önemli rol oynar. Sık erişilen düğümler ve ilişkiler bellek (RAM) üzerinde tutulur. Bu sayede gezinme işlemleri disk erişimi olmadan gerçekleştirilebilir. Özellikle sıcak veri (hot data) üzerinde yapılan sorgularda bu mekanizma performansı ciddi şekilde artırır.

Gezinme motoru, graf üzerindeki gezinme işlemlerini yöneten bileşendir. Cypher sorguları, bu motor tarafından optimize edilerek graf üzerinde adım adım yürütülür. Her adımda komşu düğümler keşfedilir ve sorgu ağ boyunca ilerler. Bu süreç, klasik sorgu motorlarındaki tablo taramasından tamamen farklıdır.

Bellek yerelliği (memory locality) Neo4j'nin bir diğer önemli avantajıdır. İlgili düğümler ve ilişkiler fiziksel olarak birbirine yakın saklandığı için bellek erişim maliyeti düşer. Bu durum, özellikle büyük graf yapılarında performansın stabil kalmasını sağlar.

Son olarak scale-up ve scale-out yaklaşımı mimari tartışmanın önemli bir parçasıdır. Neo4j geleneksel olarak scale-up (dikey ölçekleme) modelinde güçlüdür; yani tek bir güçlü sunucu üzerinde yüksek performans sağlar. Ancak modern sürümlerle birlikte scale-out (yatay ölçekleme) desteği de gelişmiştir. Bu sayede büyük graf veri setleri birden fazla düğüm arasında dağıtılarak işlenebilir.

Sonuç olarak Neo4j, yalnızca bir veri tabanı değil, graf düşüncesini fiziksel mimariye dönüştüren bir sistemdir. İlişkileri veri modelinin merkezine yerleştirerek, veri dünyasında JOIN tabanlı düşünceden gezinme tabanlı düşünceye geçişi mümkün kılar.

4.5 Cypher: Graf ile Konuşmak

Cypher, graf veri tabanlarıyla etkileşim kurmak için geliştirilmiş, bildirime dayalı bir sorgu dilidir. Özellikle Neo4j ekosistemi içinde doğmuş ve zamanla graf veri dünyasının en yaygın araçlarından biri haline gelmiştir. Cypher'ın temel amacı, karmaşık graf sorgularını mümkün olduğunca okunabilir ve sezgisel hale getirmektir. Bu nedenle tasarım felsefesi, "veriyi nasıl bulurum?" sorusunu SQL'deki gibi adım adım işlemlerle değil, "hangi deseni arıyorum?" sorusuyla cevaplamaya dayanır.

Cypher'ın doğuşu, graf veri modellerinin giderek karmaşıklaşması ve bu yapıların SQL ile ifade edilmesinin zorlaşmasıyla doğrudan ilişkilidir. Özellikle çok adımlı ilişkilerin (multi-hop relationships) SQL'de uzun JOIN zincirlerine dönüşmesi, daha doğal bir ifade biçimi ihtiyacını doğurmuştur. Cypher bu ihtiyaca cevap olarak, graf üzerindeki ilişkileri doğrudan ifade eden bir sözdizimi ile geliştirilmiştir.

Cypher'ın en dikkat çekici özelliklerinden biri ASCII-art benzeri sorgu mantığıdır. Yani sorgular, görsel olarak graf yapısını andıran bir ifade biçimiyle yazılır. Örneğin:

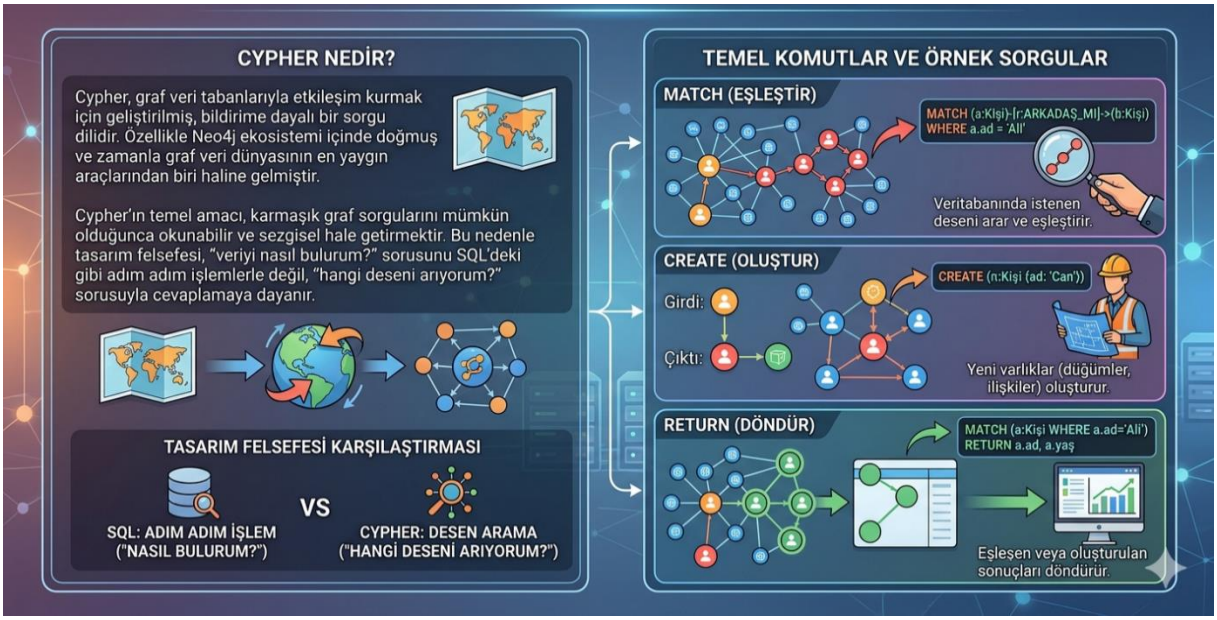
```
(a)-[:KNOWS]->(b)
```

ifadesi, iki düğüm arasındaki yönlü bir ilişkiyi doğrudan görsel olarak temsil eder. Bu yaklaşım, geliştiricinin zihnindeki graf modelini sorgu diline neredeyse birebir aktarmasına olanak tanır.

Cypher’ın merkezinde örüntü eşleştirme (pattern matching) yaklaşımı bulunur. Bu yaklaşımda kullanıcı, verinin nasıl hesaplanacağını değil, hangi yapısal örüntünün bulunacağını tanımlar. Sistem bu örüntüyü graf üzerinde eşleştirerek sonuçları üretir. Bu, SQL’deki prosedürel düşünmeden farklı olarak tamamen deklaratif bir modeldir.

Cypher’ın en temel üç komutu MATCH, CREATE ve RETURN yapılarıdır. MATCH, graf üzerinde belirli bir örüntüyü bulmak için kullanılır. CREATE, yeni düğüm ve ilişkileri grafa ekler. RETURN ise sorgu sonucunda hangi verinin döndürüleceğini belirler. Bu üçlü yapı, graf veri ile etkileşimin temelini oluşturur.

Şekil 17. Cypher Sorgulama Dili



Şekil 17, graf veri tabanları (özellikle Neo4j) ile çalışırken kullanılan en popüler ve güçlü sorgu dili olan Cypher’ı tanıtan, hem teorik mantığını hem de temel pratik komutlarını bir arada sunan harika bir eğitim şemasıdır. Şema, mantıksal olarak iki büyük ana panele ayrılmıştır: Sol taraf dilin felsefesini ve tanımını, sağ taraf ise en temel 3 komutun nasıl çalıştığını görsel örneklerle açıklar.

1. Sol Panel: Cypher Nedir ve Tasarım Felsefesi

Bu panel, kullanıcının graf sorgulama mantığına zihinsel bir geçiş yapmasını sağlar.

- Tanım: Cypher'ın bildirime dayalı (declarative) bir sorgu dili olduğu, Neo4j ekosisteminde doğup graf dünyasının standardı haline geldiği belirtiliyor.
- Temel Amacı: Karmaşık graf ilişkilerini okunabilir ve sezgisel kılmaktır. Metinde en kritik vurgu şudur: SQL gibi "Veriyi adım adım nasıl bulurum?" (prosedürel yaklaşım) sorusuna değil, "Hangi deseni (pattern) arıyorum?" sorusuna odaklanır.
- Tasarım Felsefesi Karşılaştırması (Alt Kısım): * SQL: İlişkisel veri tabanları için adım adım işlem mantığı devrededir ("Nasıl bulurum?").
    - Cypher: Graf veri tabanları için desen arama mantığı devrededir ("Hangi deseni arıyorum?"). Görselde ASCII art mantığına göz kırpan bir dünya haritası ve graf

sembolü yer alıyor; çünkü Cypher, kod yazarken parantezlerle ()--[]->() şeklinde düğüm ve ilişkileri adeta "çizer".

2. Sağ Panel: Temel Komutlar ve Örnek Sorgular

Burası Cypher dilinin kutsal üçlüsü diyebileceğimiz MATCH, CREATE ve RETURN komutlarının anatomisini çıkarır. Dikkat çekici bir detay olarak, sorgu bloklarının içindeki yazımlar ve parantez yapıları (ASCII art mantığı) teknik olarak tamamen doğrudur:

Bu şema, ilişkisel veri tabanından (SQL) graf veri tabanına geçiş yapacak bir öğrenciye veya uzmana "Cypher gibi düşünme" yeteneğini kazandırmak için tasarlanmış mükemmel bir başvuru kılavuzudur. Özellikle Bilgi Grafları mimarisinde sorguların arkasındaki mantığı çok duru bir şekilde özetlemektedir.

Patikada gezinme (path traversal) düşüncesi Cypher'ın en önemli kavramsal yeniliğidir. Burada sorgu, tablo satırları üzerinde değil, düğümler arasındaki yollar (paths) üzerinde ilerler. Bir düğümden diğerine geçiş, graf boyunca yapılan doğal bir gezinti olarak modellenir. Bu yaklaşım, özellikle çok adımlı ilişkilerde veri sorgulama sürecini sezgisel hale getirir.

> Cypher ile SQL arasındaki en temel zihinsel fark, veri modeline bakış biçiminde ortaya çıkar. SQL, veriyi tablolar ve bu tablolar arasındaki ilişkiler üzerinden düşünürken; Cypher veriyi bir ağ yapısı olarak ele alır. Bu fark yalnızca sözdizimsel değil, aynı zamanda bilişsel bir farktır. SQL'de geliştirici "hangi tabloları JOIN etmeliyim?" diye düşünürken, Cypher'da "hangi yolu takip etmeliyim?" sorusu öne çıkar.

4.5.1 SQL vs Cypher

SQL ve Cypher, aynı problemi çözmeye çalışan ancak tamamen farklı zihinsel modeller üzerine kurulu iki yaklaşımdır. SQL'in temelinde JOIN mekanizması bulunur. Veriler ayrı tablolar halinde tutulur ve aralarındaki ilişkiler sorgu anında JOIN işlemleriyle birleştirilir. Bu yaklaşım güçlüdür, ancak ilişkiler arttıkça sorgular karmaşık hale gelir.

Cypher ise JOIN yerine gezinme mantığını kullanır. Yani ilişkiler sorgu sırasında hesaplanmaz; veri modelinin içinde zaten mevcuttur. Bu nedenle sistem, bir düğümden diğerine doğrudan geçerek graf üzerinde ilerler. Bu yaklaşım, özellikle derin ve çok katmanlı ilişkilerde büyük performans ve okunabilirlik avantajı sağlar.

Tablo düşüncesi ile ağ düşüncesi arasındaki fark, bu iki yaklaşımın özünü oluşturur. SQL dünyasında veri, satır ve sütunlardan oluşan statik yapılarda organize edilir. Cypher dünyasında ise veri, sürekli genişleyen dinamik bir ağdır. Bu ağ içinde her varlık, diğer varlıklarla olan ilişkileri üzerinden anlam kazanır.

Aynı problemi bu iki yaklaşımla çözmek, çoğu zaman iki farklı zihinsel model kurmak anlamına gelir. SQL çözümü genellikle daha uzun ve teknik olarak daha prosedürel olurken, Cypher

çözümü daha kısa, daha görsel ve daha sezgisel bir yapı sunar. Özellikle çok adımlı ilişkilerde Cypher, problemi doğrudan graf üzerinde ifade ettiği için doğal bir anlatım sağlar.

Sonuç olarak SQL ve Cypher karşılaştırması yalnızca iki farklı sorgu dili karşılaştırması değildir; aynı zamanda iki farklı veri düşünme biçiminin karşılaştırılmasıdır. SQL tablo merkezli bir dünyayı temsil ederken, Cypher ilişki merkezli bir dünyayı temsil eder. Bu fark, modern veri sistemlerinde neden graf düşüncesinin giderek daha önemli hale geldiğini anlamak için kritik bir eşiktir.

4.6. **Neo4j'ye Veri Girişi**

Neo4j dünyasında veri girişi, klasik ilişkisel veri tabanı mantığından oldukça farklıdır. SQL tabanlı sistemlerde veriler satırlar ve sütunlar halinde tablolarda tutulur. Neo4j'de ise temel yapı taşları düğümler, ilişkiler ve özelliklerdir. Bu nedenle Neo4j'ye veri yüklemek yalnızca "kayıt eklemek" değil, aynı zamanda gerçek dünyanın ilişkisel yapısını modellemek anlamına gelir.

Bir ilişkisel veri tabanında şu soru sorulur:

"Bu veri hangi tabloda tutulacak?"

Neo4j'de ise soru değişir:

"Bu varlık başka hangi varlıklarla ilişkilidir?"

Bu yaklaşım özellikle:

- sosyal ağlar,
- tedarik zincirleri,
- bilgi grafı sistemleri,
- biyoinformatik,
- dolandırıcılık tespiti,
- savunma analizleri,
- öneri sistemi mimarileri
  gibi alanlarda büyük avantaj sağlar.

Neo4j'ye veri girmenin birçok yöntemi vardır. Bu bölümde en önemli yöntemleri örnek kodlarla birlikte inceleyeceğiz.

1. Cypher ile Manuel Veri Girişi

Neo4j'nin sorgu dili olan Cypher, SQL'e benzer şekilde okunabilir bir yapıya sahiptir. Ancak graf düşüncesine göre tasarlanmıştır. İlk örneğimizde bir kişi düğümü oluşturalım.

```
CREATE (p:Person {
  name:'Ahmet',
  age:35,
  city:'Bursa'
})
```

Burada:

- Person bir etiket (label),
- name, age, city ise özelliklerdir.

Oluşturulan düğüm şu anlamı taşır:

(Ahmet:Person)

Şimdi iki kişi oluşturalım:

```
CREATE (a:Person {name:'Ayşe'})
CREATE (b:Person {name:'Mehmet'})
```

Bu kişiler arasında bir ilişki kurabiliriz:

```
MATCH (a:Person {name:'Ayşe'})
MATCH (b:Person {name:'Mehmet'})
CREATE (a)-[:KNOWS]->(b)
```

Bu sorgu şu graf yapısını üretir:

Ayşe ----KNOWS----> Mehmet
İlişkiler de özellik taşıyabilir:

```
MATCH (a:Person {name:'Ayşe'})
MATCH (b:Person {name:'Mehmet'})

CREATE (a)-[:KNOWS {
   since:2020,
   closeness:'high'
}]->(b)
```

Burada artık yalnızca “tanıyor” bilgisi değil, ilişkinin niteliği de tutulmaktadır. Bu yaklaşım bilgi grafi sistemlerinin temelidir. Çünkü gerçek dünyada çoğu zaman önemli olan nesnelerin kendisi değil, aralarındaki ilişkidir.

2. Çoklu Veri Oluşturma

Gerçek projelerde binlerce düğüm oluşturulur. Cypher toplu veri girişini destekler. Örneğin:

```
CREATE
(:Person {name:'Ali'}),
(:Person {name:'Zeynep'}),
(:Person {name:'Can'}),
(:Person {name:'Elif'})
```

İlişkilerle birlikte:

```
CREATE
(a:Person {name:'Ali'}),
(b:Person {name:'Zeynep'}),
(c:Person {name:'Can'}),

(a)-[:FRIEND]->(b),
(a)-[:WORKS_WITH]->(c)
```

Bu yapı artık küçük bir sosyal ağ oluşturmaktadır.

3. CSV Dosyalarından Veri Yükleme

Gerçek dünyada veriler çoğunlukla:

- Excel,
- ERP sistemleri,
- CRM yazılımları,
- SQL veri tabanları

içinden gelir.

Bu nedenle CSV aktarımı Neo4j projelerinde en yaygın yöntemlerden biridir. Örneğin elimizde şu CSV dosyası olsun:

```
name,age,city
Ali,30,Bursa
Ayşe,28,İstanbul
Mehmet,40,Ankara
```

Dosya adı:

people.csv

Şimdi bunu Neo4j'ye aktaralım:

```
LOAD CSV WITH HEADERS
FROM 'file:///people.csv' AS row

CREATE (:Person {
   name: row.name,
   age: toInteger(row.age),
   city: row.city
})
```

Burada:

- LOAD CSV dosyayı okur,
- WITH HEADERS ilk satırın sütun adı olduğunu belirtir,
- row.name ilgili sütunu temsil eder.

toInteger() dönüşümü önemlidir. Çünkü CSV verileri başlangıçta metin olarak okunur.
İlişkilerin CSV ile yüklenmesinde ilişkiler için ayrı bir dosya kullanılır.

Örneğin:

from,to
Ali,Ayşe
Ali,Mehmet

Bu dosya:
relations.csv
olarak kaydedilsin.

Şimdi ilişkileri oluşturalım:

```
LOAD CSV WITH HEADERS
FROM 'file:///relations.csv' AS row

MATCH (a:Person {name: row.from})
MATCH (b:Person {name: row.to})

CREATE (a)-[:KNOWS]->(b)
```

Bu yöntem milyonlarca ilişkiyi yüklemek için kullanılabilir.

4. MERGE Kullanımı

CREATE her zaman yeni düğüm üretir. Ancak bazen tekrar eden kayıt istemeyiz. Örneğin aynı kişiyi iki kez yüklemek istemiyorsak:

```
MERGE (p:Person {name:'Ahmet'})
```

MERGE:
- düğüm varsa kullanır,
- yoksa oluşturur.

Kurumsal projelerde genellikle MERGE tercih edilir.

Örneğin müşteri verisi güncelleme:

```
MERGE (c:Customer {id:101})
SET c.name = 'Ayşe Yılmaz',
   c.city = 'Bursa'
```

Bu yaklaşım ETL süreçlerinde çok önemlidir.

5. Python ile Neo4j'ye Veri Girişi

Modern veri bilimi projelerinde en güçlü yöntemlerden biri Python entegrasyonudur. Neo4j Python sürücüsü kurulmalıdır:

```
pip install neo4j
```

Şimdi bağlantı oluşturalım:

```
from neo4j import GraphDatabase

driver = GraphDatabase.driver(
   "bolt://localhost:7687",
   auth=("neo4j", "password")
)
```

Veri ekleme fonksiyonu:

```
def add_person(tx, name, age):

   tx.run(
      """
      CREATE (:Person {
         name:$name,
         age:$age
      })
      """,
      name=name,
      age=age
   )
```

Kullanımı:

```
with driver.session() as session:

   session.execute_write(
      add_person,
      "Ahmet",
      35
   )
```

Bu yöntem:

- otomasyon,
- veri boru hatları (pipeline),

- AI sistemleri,
- streaming mimarileri
  için son derece uygundur.

6. Pandas DataFrame'den Veri Aktarma

Veri bilimciler çoğu zaman Pandas kullanır. Örneğin:

```
import pandas as pd

df = pd.DataFrame({
   "name":["Ali","Ayşe","Can"],
   "age":[30,28,40]
})
```

Şimdi DataFrame içeriğini Neo4j'ye aktaralım:

```
from neo4j import GraphDatabase

driver = GraphDatabase.driver(
   "bolt://localhost:7687",
   auth=("neo4j","password")
)

with driver.session() as session:

   for _, row in df.iterrows():

      session.run(
         """
         CREATE (:Person {
            name:$name,
            age:$age
         })
         """,
         name=row["name"],
         age=row["age"]
      )
```

Bu yaklaşım özellikle:

- veri temizleme,
- feature engineering,
- exploratory data analysis,
- machine learning hazırlığı
  süreçlerinde önemlidir.

7. JSON Verilerinin Aktarılması

Modern web servisleri çoğunlukla JSON üretir. Örneğin:

```
{
  "name":"Ahmet",
  "city":"Bursa",
  "skills":["Python","Neo4j","AI"]
}
```

Python ile işlenebilir:

```
import json
```

ve ardından Neo4j'ye aktarılabilir. Bu yöntem:

- REST API,
- mikroservis mimarileri,
- web uygulamaları
  için yaygındır.

8. Büyük Ölçekli Veri Aktarımı

Milyonlarca kayıt için toplu import araçları kullanılır. Neo4j'nin neo4j-admin import aracı oldukça hızlıdır. Örneğin:

```
neo4j-admin database import full
```

Bu yöntem:

- sosyal ağlar,
- telekom verileri,
- finans ağları,
- savunma sistemleri
  gibi büyük veri projelerinde kullanılır.

Çünkü Cypher ile tek tek veri eklemek milyonlarca kayıt için yavaş kalabilir.

9. Gerçek Zamanlı Veri Akışı

Modern sistemlerde veri sürekli akar. Örneğin:

- IoT sensörleri,
- banka işlemleri,
- radar sistemleri,
- sosyal medya akışları
  anlık veri üretir.

Bu nedenle Neo4j:

- Kafka,
- Spark,
- Flink,
- REST API
  gibi sistemlerle entegre çalışabilir.

Örneğin banka dolandırıcılık tespiti:

Kart ----KULLANDI----> ATM
Kart ----TRANSFER----> Hesap
Hesap ----SAHİBİ----> Kişi

şeklinde modellenebilir. Gerçek zamanlı veri akışı sayesinde şüpheli ağ yapıları anında bulunabilir.

10. Neo4j Browser ile Görsel Veri Girişi

Neo4j Browser kullanıcıların sorguları görsel olarak çalıştırmasını sağlar. Örneğin:

MATCH (n)
RETURN n

graf yapısını ekranda gösterir.

Ayrıca kullanıcılar:

- düğümlere tıklayabilir,
- ilişkileri inceleyebilir,
- veri keşfi yapabilir.
-

Bu nedenle Neo4j yalnızca veri depolama sistemi değil, aynı zamanda görsel keşif aracıdır.

11. Veri Modelleme Mantığı

Neo4j'de veri girişinden daha önemli olan konu veri modelidir. Örneğin klasik SQL yaklaşımında:

Müşteri Tablosu
Sipariş Tablosu
Ürün Tablosu
vardır.

Neo4j yaklaşımında ise:

Müşteri ----SATIN_ALDI----> Ürün
Müşteri ----VERDİ----> Sipariş
Sipariş ----İÇERİR----> Ürün
şeklinde ilişki merkezli düşünülür.

Bu yaklaşım özellikle karmaşık bağlantıları analiz etmeyi kolaylaştırır.

12. Bilgi Grafı Perspektifi

Neo4j'ye veri girişi aslında bilgi grafı oluşturma sürecidir. Her veri üç bileşenden oluşur:

Varlık + İlişki + Varlık

Örneğin:

Einstein ----GELİŞTİRDİ----> Görelilik Kuramı

Bu nedenle Neo4j yalnızca bir veri tabanı değildir. Aynı zamanda:

- anlam ilişkilerini,
- bağlamsal yapıları,
- semantik bağlantıları
  temsil eden bir düşünme modelidir.

Modern yapay zekâ sistemlerinde graf veritabanı teknolojilerinin yükselmesinin temel nedeni de budur. Çünkü gerçek dünya düz tablolar halinde değil, karmaşık ilişkiler ağı biçiminde çalışır.

4.7 Neo4j ile İlk Uygulama: Tedarik Zinciri Grafı

Neo4j'nin gerçek gücü, karmaşık ilişkiler içeren sistemlerde ortaya çıkar. Bu nedenle Neo4j öğrenmenin en iyi yollarından biri, gerçek dünyadaki ağ yapılarından birini modellemektir. Tedarik zincirleri bunun için son derece uygun örneklerdir. Çünkü modern üretim dünyası aslında birbirine bağlı fabrikalar, bileşenler, limanlar, lojistik merkezleri, şehirler ve müşterilerden oluşan dev bir ilişkiler ağıdır.
Bir otomobilin ya da akıllı telefonun üretimini düşünelim. Nihai ürün yüzlerce parçadan oluşur. Bu parçaların bir kısmı başka ülkelerde üretilir, bazıları belirli fabrikalarda monte edilir, ardından farklı şehirlere gönderilir. Küçük bir aksama bazen tüm sistemi etkileyebilir. Pandemi döneminde yaşanan çip krizi bunun klasik örneklerinden biridir. Tek bir yarı iletken bileşenindeki sorun, dünya çapındaki otomotiv üretimini durdurabilmiştir. İşte Neo4j bu tür karmaşık bağımlılık yapılarını doğal biçimde modellemek için geliştirilmiştir.

Problemin Tanımlanması

Örneğimizde elektrikli araç üreten küresel bir şirket düşünelim. Bu şirketin:

- Bursa'da montaj fabrikası,
- Berlin'de batarya fabrikası,

- Shenzhen’de elektronik bileşen üreticileri,
- Rotterdam’da lojistik merkezi,
- İstanbul, Paris ve Dubai’de dağıtım noktaları olsun.

Üretilen araç şu temel bileşenlerden oluşsun:

- Lityum batarya
- Motor kontrol ünitesi
- Yapay zekâ sürüş modülü
- Sensör sistemi

Bu yapıda kritik soru şudur:

“Bir bileşende yaşanan sorun tüm sistemi nasıl etkiler?”

Klasik SQL yaklaşımında bu sorunun cevabı onlarca JOIN işlemi gerektirebilir. Neo4j’de ise sistem doğrudan ağ olarak modellenir.

Graf Modelinin Tasarımı

İlk adım düğüm tiplerini belirlemektir.

Product düğümü

```
CREATE (:Product {
   name:'Electric SUV',
   price:85000
})
```

Bu sorgunun anlamı şudur:
Bir Product düğümü oluştur.
Adı Electric SUV olsun.
Fiyatı 85.000 olsun.

Neo4j’nin cevabı:

Node created with label Product

Properties:

name = Electric SUV
price = 85000

Şimdi bileşenleri oluşturalım.

```
CREATE (:Component {
   name:'Lithium Battery',
```

```
  criticality:'high'
})

CREATE (:Component {
  name:'AI Driving Module',
  criticality:'medium'
})
```

Burada sistem artık yalnızca veri değil, anlam da taşımaktadır. Çünkü “criticality” özelliği bileşenin risk derecesini belirtmektedir.

Şimdi fabrikaları oluşturalım:

```
CREATE (:Factory {
  name:'Bursa Plant',
  country:'Türkiye'
})

CREATE (:Factory {
  name:'Berlin Battery Center',
  country:'Germany'
})
```

Şehir düğümleri:

```
CREATE (:City {name:'Istanbul'})
CREATE (:City {name:'Paris'})
CREATE (:City {name:'Dubai'})
```

İlişkilerin Kurulması

Şimdi sistem anlamlı hale gelmeye başlar.

```
Bir ürün hangi bileşenleri içeriyor?
MATCH (p:Product {name:'Electric SUV'})
MATCH (c:Component {name:'Lithium Battery'})

CREATE (p)-[:CONTAINS]->(c)
```

Bu sorgunun doğal dil karşılığı:

“Electric SUV ürünü Lithium Battery bileşenini içeriyor.”

Neo4j cevabı:

```
Relationship created:
(Electric SUV)-[:CONTAINS]->(Lithium Battery)
```

Şimdi üretim ilişkisini tanımlayalım:

```
MATCH (p:Product {name:'Electric SUV'})
MATCH (f:Factory {name:'Bursa Plant'})

CREATE (p)-[:MANUFACTURED_AT]->(f)
```

Bu ilişki artık şu anlama gelir:

Electric SUV Bursa Plant fabrikasında üretiliyor.

Lojistik Ağının Oluşturulması

Neo4j'nin en güçlü yönlerinden biri ilişkilerin özellik taşıyabilmesidir. Örneğin:

```
MATCH (f:Factory {name:'Bursa Plant'})
MATCH (c:City {name:'Paris'})

CREATE (f)-[:SHIPPED_TO {
   cost:1200,
   time:4,
   capacity:500
}]->(c)
```

Burada ilişki yalnızca bağlantı değildir.

Aynı zamanda:

- maliyet,
- süre,
- kapasite

bilgilerini de taşımaktadır.
Bu nedenle graf artık yalnızca bağlantı haritası değil, aynı zamanda operasyonel bilgi sistemi haline gelir.

İlk MATCH Sorguları

Graf oluşturulduktan sonra en heyecanlı aşama başlar: gezinme (traversal). Örneğin şu soruyu soralım:

"Electric SUV hangi bileşenleri içeriyor?"

Cypher sorgusu:

```
MATCH (p:Product {name:'Electric SUV'})
    -[:CONTAINS]->
    (c:Component)

RETURN c.name
```

Neo4j cevabı:

Lithium Battery
AI Driving Module

Burada yapılan şey şudur:

- Product düğümünden başlanır,
- CONTAINS ilişkisi takip edilir,
- Component düğümlerine gidilir.

Bu tam anlamıyla ağ üzerinde gezinmedir.

Çok Adımlı Gezinme

Şimdi daha karmaşık bir soru soralım:

“Electric SUV hangi şehirlerde dağıtılıyor?”

```
MATCH (p:Product {name:'Electric SUV'})
    -[:MANUFACTURED_AT]->
    (f:Factory)
    -[:SHIPPED_TO]->
    (c:City)

RETURN c.name
```

Neo4j’nin cevabı:
Paris
Dubai
Istanbul

Bu sorgunun güzelliği şudur:

Kod neredeyse doğal dil gibidir. Aslında sorgu şunu söylemektedir:

“Electric SUV ürününden başla, onu üreten fabrikaya git, ardından gönderim yapılan şehirleri bul.”

Bu nedenle Cypher çoğu zaman SQL’den daha sezgiseldir.

İlişki Filtreleme ile lojistik maliyetleri analizi

Soru:

“Maliyeti 1000 doların altında olan gönderim rotaları hangileri?”

```
MATCH (f:Factory)-[s:SHIPPED_TO]->(c:City)

WHERE s.cost < 1000

RETURN f.name,
    c.name,
    s.cost
```

Neo4j cevabı:

Bursa Plant → Istanbul → 700
Berlin Battery Center → Paris → 900

Burada dikkat edilmesi gereken nokta:

- filtreleme düğüm üzerinde değil,
- ilişkinin özelliği üzerinde yapılmaktadır.

Bu, graph database yaklaşımının temel farklarından biridir.

Etki Analizi (Impact Analysis)

Şimdi en önemli uygulamaya gelelim.

Soru:

“Lithium Battery bileşeninde sorun çıkarsa hangi ürünler, fabrikalar ve şehirler etkilenir?”

Cypher sorgusu:

```
MATCH (c:Component {name:'Lithium Battery'})
< -[:CONTAINS]-
(p:Product)
-[:MANUFACTURED_AT]->
(f:Factory)
-[:SHIPPED_TO]->
(city:City)

RETURN p.name,
    f.name,
    city.name
```

Neo4j cevabı:

Electric SUV | Bursa Plant | Paris
Electric SUV | Bursa Plant | Dubai
Electric SUV | Bursa Plant | Istanbul

Bu sorgu aslında bir downstream etki analizidir.

Neo4j burada:

- bileşenden ürüne,
- üründen fabrikaya,
- fabrikadan şehirlere

doğru ilerleyerek tüm bağımlılık ağını ortaya çıkarmaktadır. Klasik ilişkisel sistemlerde bu tür analizler oldukça karmaşık hale gelir.

Risk Yayılımı

Graf yaklaşımının en önemli avantajlarından biri zincirleme etkileri görünür hale getirmesidir.

Örneğin:

Lithium Battery
↓
Electric SUV
↓
Bursa Plant
↓
Paris / Dubai / Istanbul

Bu yapı aslında bir "etki dalgası"dır. Savunma sistemleri, finans ağları ve kritik altyapılar tam olarak bu mantıkla analiz edilir.

En Kısa Yol Analizi

Şimdi lojistik optimizasyonuna geçelim.

Soru:

"Berlin'den Dubai'ye en düşük maliyetli rota nedir?"

Neo4j'de ilişkiler ağırlık taşıyabildiği için bu tür analizler mümkündür. Örneğin SHIPPED_TO ilişkilerindeki cost özelliği kullanılabilir.

Cypher:

```
MATCH path =
shortestPath(
(:Factory {name:'Berlin Battery Center'})
-[*]->
(:City {name:'Dubai'})
)

RETURN path
```

Neo4j cevabı:

Berlin Battery Center
  ↓
Rotterdam Hub
  ↓
Dubai

Daha gelişmiş senaryolarda:

- Dijkstra,
- A*,
- PageRank,
- Community Detection

gibi algoritmalar da çalıştırılabilir.

Bu noktada Neo4j yalnızca veri tabanı olmaktan çıkar ve bir ağ analitiği platformuna dönüşür.

SQL ile Aynı Problemin Zorluğu

Aynı sorgunun SQL dünyasında nasıl görüneceğini düşünelim.

Bir bileşenin etkilediği şehirleri bulmak için:

- Components tablosu,
- Products tablosu,
- Factories tablosu,
- Logistics tablosu,
- Cities tablosu

arasında çok sayıda JOIN gerekir.
Örneğin:

```
SELECT ...
FROM Components
JOIN Products ...
JOIN Factories ...
```

JOIN Logistics ...
JOIN Cities ...

JOIN sayısı arttıkça:

- sorgu okunamaz hale gelir,
- performans düşer,
- bakım zorlaşır.

Neo4j'de ise ağ zaten fiziksel olarak ilişki biçiminde tutulmaktadır. Bu nedenle gezinme işlemleri çok daha doğaldır.

Neden Bu Yaklaşım Güçlüdür?

Neo4j'nin temel gücü şuradan gelir: Gerçek dünya zaten graf yapısındadır.

İnsanlar,
ürünler,
şehirler,
tedarikçiler,
bankalar,
sensörler,
sosyal ağlar…

hepsi birbirine bağlıdır.

Graf veri tabanı bu bağlantıları sonradan hesaplamaz; doğrudan veri modelinin merkezine yerleştirir.

Bu nedenle:

- ilişkiler birinci sınıf vatandaştır,
- gezinme hızlıdır,
- bağımlılıklar görünürdür,
- ağ düşüncesi doğal hale gelir.

Tedarik zinciri örneği bunu açık biçimde göstermektedir. Neo4j yalnızca "veri saklayan" bir sistem değildir. Aynı zamanda karmaşık dünyayı ağlar üzerinden düşünmeyi sağlayan yeni bir bilişsel modeldir.

4.7 Neo4j ile İlk Uygulama: Tedarik Zinciri Grafı

Bu bölümde Neo4j'nin en güçlü yönünü görmek için gerçekçi bir uygulama senaryosu üzerinden ilerleyeceğiz: tedarik zinciri ve lojistik ağı. Bu tür sistemler doğal olarak graf yapısındadır çünkü ürünler, bileşenler, fabrikalar ve şehirler arasında çok katmanlı ve sürekli değişen ilişkiler vardır. Amaç, bu karmaşık yapıyı klasik tablo mantığıyla değil, doğrudan graf modeliyle temsil etmek ve analiz etmektir.

Problemin Tanımlanması

Bir şirketin küresel ölçekte ürün ürettiğini ve dağıttığını düşünelim. Bu sistemin içinde birden fazla unsur vardır:

- Ürünler (Products): Son kullanıcıya sunulan nihai çıktılar
- Bileşenler (Components): Ürünleri oluşturan parçalar
- Fabrikalar (Factories): Üretimin gerçekleştiği yerler
- Şehirler (Cities): Dağıtım noktaları
- Lojistik ağları: Fabrika ve şehirler arasındaki taşıma bağlantıları
- Risk bağımlılıkları: Bir bileşen veya fabrika aksadığında tüm sistemin etkilenmesi

Bu yapı ilk bakışta basit görünse de aslında çok katmanlı bir bağımlılık ağıdır. Örneğin bir bileşenin üretiminde yaşanan sorun yalnızca bir ürünü değil, o ürünü üreten fabrikayı, o fabrikanın hizmet verdiği şehirleri ve hatta o şehirlerdeki talep zincirini etkileyebilir. Bu nedenle sistem yalnızca veri değil, bir etki ağıdır.

Graf Modelinin Tasarımı

Bu problemi graf veri modeli ile ifade etmek için dört temel düğüm tipi tanımlarız:

- Product node: Ürünleri temsil eder
- Component node: Ürün bileşenlerini temsil eder
- Factory node: Üretim noktalarını temsil eder
- City node: Dağıtım yapılan şehirleri temsil eder

Bu düğümler arasındaki ilişkiler ise sistemi anlamlı hale getirir:

- CONTAINS: Bir ürün hangi bileşenleri içerir
- MANUFACTURED_AT: Bir ürün hangi fabrikada üretilir
- SHIPPED_TO: Fabrikadan hangi şehre gönderim yapılır

Bu modelde dikkat edilmesi gereken kritik nokta şudur: ilişkiler yalnızca bağlantı değildir, aynı zamanda anlam taşıyan veri elemanlarıdır. Örneğin SHIPPED_TO ilişkisine maliyet (cost), süre (time) veya kapasite (capacity) gibi özellikler eklenebilir. Böylece graf yalnızca “kim kime bağlı” sorusunu değil, “bu bağlantı ne kadar maliyetli veya ne kadar kritik” sorusunu da cevaplar.

Cypher ile Grafın İnşası

Neo4j’de graf yapısı Cypher dili ile oluşturulur. Örneğin:

- Product, Component, Factory ve City node’ları CREATE komutuyla oluşturulur.
- İlişkiler düğümler arasında doğrudan tanımlanır.

Bu yaklaşımın en önemli özelliği şudur: veri ve yapı aynı anda inşa edilir. Yani önce tablo oluşturup sonra veri eklemek gibi bir ayrım yoktur. Örneğin bir SHIPPED_TO ilişkisine maliyet

özelliği eklenebilir. Bu özellik, lojistik analizlerinde kritik bir rol oynar çünkü sadece bağlantının varlığını değil, aynı zamanda ekonomik ağırlığını da temsil eder. Bu noktada graf modelinin gücü ortaya çıkar: veri sadece varlıkları değil, ilişkilerin doğasını da içerir.

İlk Gezinme Sorguları

Graf oluşturulduktan sonra artık veriyi sorgulamak mümkündür. Bu noktada MATCH ifadesi devreye girer. MATCH, graf üzerinde belirli bir deseni bulmak için kullanılır. Örneğin MATCH ile bir bileşenden başlayarak hangi ürünlere ve hangi şehirlere ulaşıldığını görmek mümkündür. Bu süreç gezinme olarak adlandırılır. Yani sorgu, graf üzerinde düğümden düğüme ilerleyerek veriyi keşfeder. İlişki filtreleme (relationship filtering) ise bu gezinme sürecini daraltmak için kullanılır. Örneğin yalnızca belirli bir maliyetin altındaki lojistik bağlantılar incelenebilir.

Etki Analizi

Graf modelinin en güçlü kullanım alanlarından biri etki analizidir. Örneğin şu soruyu ele alalım: "Lityum batarya tedarikinde bir sorun olursa sistem nasıl etkilenir?" Bu durumda graf üzerinde geriye doğru bir gezinme yapılır:

Component → Product → Factory → City

Bu zincir sayesinde bir bileşendeki sorun, hangi ürünleri, hangi fabrikaları ve hangi şehirleri etkiliyor net şekilde görülebilir. Bu tür analizlere downstream etki analizi denir. Çünkü bir noktadan başlayan sorun, aşağı akıştaki tüm sistemleri etkiler. Bu yaklaşım risk görünürlüğünü artırır. Klasik sistemlerde bu tür bağımlılıklar tablo JOIN'leri ile zor hesaplanırken, graf modelinde doğal olarak görünür hale gelir.

En Kısa Yol ve Maliyet Analizi

Tedarik zinciri sistemlerinde önemli sorulardan biri de şudur:

"Bir fabrikadan bir şehre en düşük maliyetli yol nedir?"

Bu problem en kısa yol problemi olarak bilinir. Neo4j bu tür analizleri doğrudan destekler çünkü ilişkiler ağırlık (weight) taşıyabilir. Örneğin SHIPPED_TO ilişkisine maliyet özelliği eklenerek her bağlantının maliyeti tanımlanabilir.

Bu durumda Dijkstra algoritması[10] gibi klasik graf algoritmaları graf üzerinde çalıştırılabilir. Sonuç olarak sistem yalnızca bağlantıları değil, en optimal rotayı da hesaplayabilir. Bu özellik özellikle lojistik optimizasyon, rota planlama ve maliyet minimizasyonu gibi alanlarda kritik öneme sahiptir.

[10] Dijkstra algoritması, bir graf (ağ) üzerindeki belirli bir başlangıç düğümünden diğer tüm düğümlere olan en kısa yolları (en düşük maliyetli yolları) bulmaya yarayan, bilgisayar bilimlerinin en temel ve meşhur algoritmalardan biridir. 1956 yılında Hollandalı bilgisayar bilimci Edsger W. Dijkstra tarafından geliştirilmiştir.

SQL ile Aynı Problemin Çözümü

Aynı problem ilişkisel veri tabanı ile çözülmek istendiğinde durum oldukça farklıdır. SQL dünyasında bu yapı JOIN zincirleri ile ifade edilir.

Component → Product → Factory → City bağlantısı için birden fazla tablo JOIN edilmesi gerekir. Bu durum üç temel probleme yol açar:

- Karmaşıklık artar: JOIN sayısı arttıkça sorgu okunamaz hale gelir
- Performans düşer: Her JOIN ek maliyet yaratır
- Bakım zorlaşır: Sorgular büyüdükçe yönetilmesi güçleşir

Cypher yaklaşımında ise bu zincir doğrudan graf üzerinde ifade edilir. Bu nedenle sorgu hem daha kısa hem de daha sezgiseldir.

Neden Graf Yaklaşımı Güçlüdür?

Graf yaklaşımının gücü birkaç temel prensibe dayanır:

Doğallık: Gerçek dünya zaten bir ağdır. İnsanlar, ürünler ve süreçler birbirine bağlıdır. Graf modeli bu yapıyı doğrudan temsil eder.

Esneklik: Yeni düğümler ve ilişkiler eklemek mevcut yapıyı bozmaz. Sistem organik olarak büyür.

Gezinme performansı: İlişkiler fiziksel olarak saklandığı için çok adımlı sorgular hızlı çalışır. İlişki merkezli analitik: Değer yalnızca varlıkta değil, varlıklar arasındaki ilişkilerde ortaya çıkar.
Ağ düşüncesi: Veri artık tablo değil, yaşayan bir sistem olarak görülür.

Sonuç olarak bu örnek, Neo4j'nin yalnızca bir veritabanı olmadığını gösterir. Aynı zamanda veri ile düşünme biçimini değiştiren bir modeldir. Tedarik zinciri örneği üzerinden görüldüğü gibi, graf yaklaşımı karmaşık sistemleri hem daha doğal hem de daha analitik biçimde temsil etmemizi sağlar.

4.8 Graf Algoritmaları ve Analitik

Graf veri tabanları çoğu zaman yalnızca bir "depolama modeli" olarak yanlış anlaşılır. Oysa graf yapılarının asıl gücü, veriyi saklamaktan çok analiz edebilmesidir. Çünkü graf, doğası gereği ilişkisel bir hesaplama alanıdır. Bu nedenle modern graf sistemleri yalnızca veri tutmaz; aynı zamanda bu veri üzerinde çalışan algoritmaları da destekler. Bu yaklaşım "Graf Veri Bilim" (Graph Data Science) olarak adlandırılır ve veriyi bir ağ olarak görüp bu ağ üzerinde matematiksel ve istatistiksel analizler yapmayı mümkün kılar.

Graf veri bilim yaklaşımı, klasik veri bilimi yöntemlerinden farklıdır. Geleneksel analizde veri genellikle satır-sütun formatında işlenir ve özellikler üzerinden modelleme yapılır. Graf yaklaşımında ise özelliklerin yanı sıra ilişkilerin yapısı da analiz edilir. Yani yalnızca "ne var?"

değil, “neyle bağlı?” sorusu da modele dahil edilir. Bu, özellikle karmaşık sistemlerde çok daha güçlü öngörüler üretir.

Neo4j Graph Data Science Library, bu yaklaşımın en gelişmiş uygulama araçlarından biridir. Bu kütüphane, graf üzerinde çalışan yüzlerce algoritmayı doğrudan veri tabanı üzerinde çalıştırmayı mümkün kılar. Böylece veri dışarı çıkarılmadan, sistem içinde analiz edilir. Bu durum hem performansı artırır hem de veri bütünlüğünü korur.

Merkeziyet (Centrality)

Merkeziyet analizleri, graf içindeki en önemli düğümleri belirlemek için kullanılır. Bu analizler, özellikle ağ içindeki kritik noktaları anlamak açısından büyük önem taşır. Derece merkeziliği, bir düğümün doğrudan kaç bağlantıya sahip olduğunu ölçer. Yüksek derece merkeziliği değeri, o düğümün ağ içinde yüksek etkileşimde olduğunu gösterir. Örneğin bir tedarik zincirinde çok sayıda ürüne hizmet veren bir tedarikçi, yüksek derece merkeziliği değerine sahip olabilir.

Arasındalık merkeziliği ise bir düğümün diğer düğümler arasındaki en kısa yollar üzerinde ne kadar sık yer aldığını ölçer. Bu metrik, ağ içindeki “köprü” noktaları belirlemek için kullanılır. Örneğin lojistik ağında birçok rota üzerinden geçen bir dağıtım merkezi, yüksek arasındalık merkeziliği değerine sahip olabilir.

Bu analizlerin en kritik kullanım alanlarından biri kritik tedarikçi analizidir. Bir sistemde hangi tedarikçinin aksaması durumunda en büyük etkiyi yaratacağı, merkeziyet metrikleri sayesinde belirlenebilir. Bu sayede risk yönetimi daha proaktif hale gelir.

Topluluk Belirleme

Topluluk belirleme (Community detection), graf içindeki yoğun şekilde birbirine bağlı düğüm gruplarını bulmayı amaçlar. Bu gruplar genellikle doğal kümeler oluşturur. Lojistik sistemlerde bu analiz, benzer ürünlerin veya aynı bölgeye hizmet eden tedarik ağlarının belirlenmesini sağlar. Örneğin belirli fabrikalar ve şehirler arasında yoğun bağlantı varsa, bu bir lojistik küme olarak tanımlanabilir.

Organizasyonel ağlarda ise bu yöntem, şirket içindeki alt yapıları ortaya çıkarır. Hangi departmanların birbirleriyle daha yoğun etkileşimde olduğu veya hangi ekiplerin aynı iş süreçleri içinde yer aldığı bu analizle görülebilir. Bu sayede organizasyonel yapı daha net anlaşılır.

Benzerlik Algoritmaları

Benzerlik (similarity) algoritmaları, graf içindeki düğümlerin birbirine ne kadar benzediğini ölçer. Bu benzerlik yalnızca özelliklere değil, aynı zamanda bağlantı yapısına da dayanabilir. Benzer ürünlerin bulunması, bu algoritmaların en yaygın kullanım alanlarından biridir. Örneğin aynı bileşenleri kullanan veya aynı fabrikalarda üretilen ürünler birbirine benzer kabul edilebilir. Öneri sistemleri de bu yaklaşım üzerine kuruludur. Bir kullanıcının daha önce etkileşimde bulunduğu ürünlere benzer ürünler önerilebilir. Bu yöntem, e-ticaret ve içerik platformlarında yaygın olarak kullanılır.

Patika Bulma Algoritmaları

Patika bulma (Pathfinding) algoritmaları, graf üzerindeki iki düğüm arasındaki en uygun yolu bulmak için kullanılır. Bu, özellikle lojistik ve ağ optimizasyonu için kritik bir alandır. Dijkstra algoritması, ağırlıklı graflarda en kısa yolu bulmak için kullanılan klasik bir algoritmadır. Her ilişkiye bir maliyet atandığında, sistem en düşük toplam maliyete sahip yolu hesaplayabilir.

A* algoritması ise Dijkstra'nın geliştirilmiş bir versiyonudur. Hem gerçek maliyeti hem de hedefe olan tahmini mesafeyi dikkate alarak daha hızlı sonuç üretir. Minimum cost routing (minimum maliyetli yönlendirme), özellikle lojistik sistemlerde en düşük maliyetli taşıma rotalarının belirlenmesini sağlar. Bu sayede hem zaman hem de maliyet optimizasyonu yapılabilir.

Dolandırıcılık ve Anomali Analizi

Graf analitiğinin en güçlü kullanım alanlarından biri de dolandırıcılık (fraud) ve anomali tespitidir. Çünkü birçok dolandırıcılık modeli doğası gereği ağ yapıları içinde gizlenir. Şüpheli işlem ağları, genellikle normal kullanıcı davranışından farklı bağlantı desenleri gösterir. Örneğin aynı hesaplar arasında hızlı ve tekrarlayan transferler olması, bir anomali göstergesi olabilir.

Döngüsel para hareketleri ise özellikle finansal dolandırıcılıkta sık görülen bir yapıdır. Para bir hesaplar arasında dolaşıp tekrar başlangıç noktasına döndüğünde, bu durum graf üzerinde döngü olarak tespit edilebilir. Bu tür analizler klasik tablo tabanlı sistemlerde oldukça zorken, graf yapılarında doğal olarak ortaya çıkar. Çünkü ilişkiler doğrudan veri modelinin içinde yer alır ve algoritmalar bu ilişkiler üzerinde çalışır.

Sonuç olarak graf algoritmaları ve analitik yaklaşımı, graf veri tabanlarını yalnızca bir veri saklama sistemi olmaktan çıkarıp güçlü bir analiz motoruna dönüştürür. Merkeziyet, topluluk tespiti, benzerlik, yol bulma ve anomali analizleri gibi yöntemler sayesinde verinin yalnızca yapısı değil, davranışı da anlaşılabilir hale gelir. Bu da modern veri biliminin en önemli dönüşümlerinden birini temsil eder: veriyi tekil nesneler olarak değil, etkileşimli bir ağ olarak analiz etmek.

4.9 Neo4j ve Bilgi Grafı Dünyası

Neo4j gibi graf veri tabanları çoğu zaman operasyonel sistemler için kullanılır: tedarik zincirleri, finansal işlemler, ağ analizleri, öneri sistemleri. Ancak graf modelinin asıl potansiyeli yalnızca operasyonel veri yönetimi değildir. Daha derin bir seviyede, bu sistemler "bilgi grafı" haline dönüşebilir. Bu dönüşüm, verinin yalnızca yapısal değil, anlamsal bir varlığa evrilmesi anlamına gelir.

Neo4j bir bilgi grafı olabilir mi?

Temel soru şudur: Neo4j bir bilgi grafı oluşturmak için yeterli midir? Cevap: tek başına değil, ancak güçlü bir altyapı olarak evet. Neo4j doğal olarak düğüm ve ilişki tabanlı bir yapı sunduğu için bilgi grafı modellemeye çok uygundur. Ancak bir bilgi grafı yalnızca veri yapısı değildir; aynı

zamanda anlam katmanı içerir. Bu nedenle Neo4j'nin üzerine ek bir anlamsal katman inşa edilmesi gerekir.

Bilgi grafı, varlıkları ve bu varlıklar arasındaki ilişkileri yalnızca bağlantı düzeyinde değil, anlam düzeyinde de temsil eder. Örneğin "bir ürün fabrikada üretilir" bilgisi yalnızca bir ilişki değildir; aynı zamanda üretim süreci, bağımlılık yapısı ve kurumsal bağlamı da içerir. Neo4j bu yapıyı temsil edebilir, ancak anlam katmanı dışarıdan eklenmelidir.

Semantik katmanın eklenmesi

Semantik katman, grafın üzerine inşa edilen anlam seviyesidir. Bu katman sayesinde düğümler ve ilişkiler yalnızca veri noktaları değil, tanımlı kavramlar haline gelir. Örneğin "Factory", "Product" veya "City" gibi düğümler artık yalnızca etiket değil, belirli bir kavramsal modelin parçalarıdır.

Bu katman sayesinde graf, "ne var?" sorusundan "bu ne anlama geliyor?" sorusuna geçer. Semantik katman olmadan graf, güçlü bir veri yapısıdır; semantik katmanla birlikte ise bir bilgi sistemine dönüşür.

Ontoloji bağlantısı

Ontoloji, bir alanın kavramsal iskeletini tanımlar. Hangi varlık türlerinin bulunduğu, bu varlıkların hangi ilişkilerle birbirine bağlanabileceği ve bu ilişkilerin hangi kurallara tabi olduğu ontoloji ile belirlenir.

Neo4j tabanlı sistemlerde ontoloji, genellikle dış bir modelleme katmanı olarak eklenir. Örneğin bir tedarik zinciri ontolojisi şu kuralları içerebilir:

- Bir ürün bir veya daha fazla bileşenden oluşur
- Bir bileşen birden fazla üründe kullanılabilir
- Bir fabrika birden fazla ürünü üretebilir
- Bir şehir birden fazla lojistik rota ile bağlanabilir

Bu tür kurallar, grafın yalnızca veri değil, aynı zamanda kurallı bir bilgi sistemi olmasını sağlar.

Varlık Eşleştirme

Bilgi grafı sistemlerinin en kritik problemlerinden biri varlık eşleştirme problemidir. Bu problem, aynı gerçek dünya varlığının farklı veri kaynaklarında farklı isimlerle veya formatlarla temsil edilmesinden kaynaklanır.

Örneğin:

- "IBM"
- "International Business Machines"
- "I.B.M."

aslında aynı şirketi temsil edebilir. Eğer bu varlıklar doğru şekilde birleştirilmezse graf içinde tutarsızlık oluşur. Varlık eşleştirme süreci, bu farklı temsil biçimlerini tek bir düğüm altında birleştirmeyi amaçlar. Bu işlem doğru yapıldığında, graf daha tutarlı ve güvenilir bir bilgi yapısına dönüşür.

Kurumsal bilgi grafı yaklaşımı

Kurumsal bilgi grafı (enterprise knowledge graph), bir organizasyonun tüm veri varlıklarını tek bir semantik yapı içinde birleştirmeyi amaçlar. Bu yapı içinde müşteri verileri, ürün bilgileri, tedarik zinciri, finansal işlemler ve operasyonel veriler aynı graf içinde temsil edilir.

Neo4j bu yaklaşım için güçlü bir altyapı sunar çünkü farklı veri kaynaklarını tek bir ilişki ağı içinde birleştirmeye izin verir. Böylece kurum içinde parçalı veri yapıları yerine bütünleşik bir bilgi sistemi oluşur. Kurumsal bilgi grafının en önemli avantajı, veri silolarını ortadan kaldırmasıdır. Farklı departmanlarda bulunan veriler artık birbirinden bağımsız değil, aynı ağın parçaları haline gelir.

Operasyonel graf → bilgi grafı dönüşümü

Neo4j gibi sistemler genellikle operasyonel graf olarak başlar. Yani amaç, sistemleri modellemek, süreçleri yönetmek ve veriyi hızlı sorgulamaktır. Ancak zamanla bu yapı daha derin bir dönüşüm geçirerek bilgi grafına evrilebilir. Bu dönüşüm üç aşamada gerçekleşir:

1. Operasyonel katman: Veri depolanır ve temel ilişkiler kurulur
2. Yapısal katman: Veri modelleri standartlaştırılır ve genişletilir
3. Semantik katman: Ontoloji ve anlam yapıları eklenir

Bu üç aşama tamamlandığında sistem artık yalnızca veri işleyen bir yapı değil, kurumsal bilgi üreten bir altyapı haline gelir. Sonuç olarak Neo4j, doğru tasarlandığında sadece bir graf veri tabanı değil, aynı zamanda bir bilgi grafı motoruna dönüşebilir. Bu dönüşüm, verinin teknik bir kaynak olmaktan çıkıp kurumsal bilginin merkezi haline gelmesini sağlar.

4.10 Neo4j + Yapay Zekâ

Yapay zekâ modellerinin özellikle büyük dil modelleri (LLM'ler) alanında ulaştığı başarı, bilgi işleme biçimimizi kökten değiştirmiştir. Ancak bu modellerin güçlü üretim kabiliyeti, bazı yapısal sınırlamalarla birlikte gelir. Bu sınırlamaların önemli bir kısmı, graf veri tabanları ve özellikle Neo4j gibi sistemlerle birlikte çalışıldığında önemli ölçüde aşılabilir. Bu bölümde, Neo4j ile yapay zekâ entegrasyonunun neden kritik olduğu ve nasıl yeni bir "bilgi mimarisi" oluşturduğu ele alınacaktır.

LLM'lerin Sınırları

Büyük dil modelleri metin üretme, özetleme ve soru yanıtlama gibi görevlerde son derece başarılıdır. Ancak bu modellerin üç temel yapısal sınırı vardır: halüsinasyon problemi, bağlam penceresi (context window) sınırı ve yapısal bilgi eksikliği. Halüsinasyon problemi, modelin gerçekte var olmayan bilgileri doğruymuş gibi üretmesidir. Bunun nedeni, LLM'lerin bilgiye

doğrudan erişen veri tabanları değil, olasılıksal metin üretim sistemleri olmasıdır. Model, doğru bilgiyi "hatırlamaz"; en olası metni üretir. Bu durum özellikle kurumsal ve teknik uygulamalarda ciddi riskler oluşturur.

İkinci önemli sınırlama bağlam penceresi sınırıdır. LLM'ler aynı anda yalnızca belirli miktarda metni işleyebilir. Bu sınır aşıldığında model önceki bilgileri unutmaya başlar. Büyük ve karmaşık sistemlerde bu durum, uzun vadeli ilişkilerin analiz edilmesini zorlaştırır.

Üçüncü sınırlama ise yapısal bilgi eksikliğidir. LLM'ler metin temelli modellerdir; dolayısıyla veriyi doğal olarak ağ yapıları, ilişkiler veya graf yapıları şeklinde temsil etmezler. Oysa gerçek dünya problemlerinin büyük kısmı yapısal ilişkiler içerir: tedarik zincirleri, finansal ağlar, sosyal ağlar ve bilgi sistemleri gibi.

Bu üç sınırlama, LLM'lerin tek başına karmaşık sistemleri anlamakta yetersiz kalmasına neden olur. İşte bu noktada graf veri tabanları devreye girer.

GraphRAG

GraphRAG (Graph Retrieval-Augmented Generation), Retrieval-Augmented Generation (RAG) yaklaşımının graf tabanlı genişletilmiş bir versiyonudur. Klasik RAG sistemlerinde model, bir soru geldiğinde ilgili metin parçalarını arar ve bunları bağlam olarak kullanır. Ancak bu yaklaşım metin tabanlıdır ve ilişkisel yapıyı tam olarak temsil edemez.

GraphRAG yaklaşımında ise bilgi erişimi süreci bir graf üzerinde gerçekleşir. Yani sistem yalnızca metin parçalarını değil, düğümler ve ilişkiler üzerinden bağlantılı bilgiyi getirir. Bu durum özellikle çok adımlı sorgularda büyük avantaj sağlar.

"Neden graph retrieval?" sorusunun cevabı burada ortaya çıkar. Çünkü gerçek dünya bilgisi yalnızca belgelerde değil, belgeler arasındaki ilişkilerde de saklıdır. Bir şirketin tedarik zinciri, ürünleri, çalışanları ve süreçleri yalnızca ayrı metinlerde değil, bu unsurlar arasındaki bağlantılarda anlam kazanır.

İlişkisel bağlam, GraphRAG sistemlerinin en önemli gücüdür. Bir bilgi parçası tek başına değil, bağlı olduğu diğer bilgilerle birlikte anlam kazanır. Bu sayede model, yalnızca metin benzerliği değil, yapı benzerliği üzerinden de sonuç üretir.

Neo4j + LLM Entegrasyonu

Neo4j ile büyük dil modellerinin entegrasyonu, yapay zekânın "veriye bağlı düşünme" yeteneğini önemli ölçüde artırır. Bu entegrasyon genellikle LangChain gibi çerçeveler üzerinden gerçekleştirilir.

LangChain, LLM'leri dış veri kaynaklarıyla birleştirmeyi sağlayan bir araçtır. Neo4j entegrasyonu sayesinde model, doğal dil sorgularını Cypher sorgularına dönüştürebilir ve graf üzerinde gerçek zamanlı sorgular yapabilir.

GraphCypherQAChain, bu yaklaşımın özel bir uygulamasıdır. Kullanıcı doğal dilde bir soru sorar ve sistem bunu Cypher sorgusuna çevirerek Neo4j üzerinde çalıştırır. Sonuç tekrar doğal dile çevrilerek kullanıcıya sunulur. Bu yapı, doğal dil ile graf verisi arasında bir köprü oluşturur.

Daha ileri bir seviyede agentic YZ sistemleri devreye girer. Bu sistemlerde LLM yalnızca cevap üreten bir model değil, aynı zamanda karar veren ve araç kullanan bir ajan haline gelir. Neo4j burada bir "araç" olarak kullanılır. Model gerektiğinde graf sorguları çalıştırır, veri çeker ve bu veriyi akıl yürütme sürecine dahil eder.

Bellek Katmanı Olarak Graf

Neo4j'nin yapay zekâ sistemlerindeki en önemli rollerinden biri bellek katmanı olarak kullanılmasıdır. LLM'ler doğası gereği durum tutmayan sistemlerdir. Her sorgu bağımsızdır ve geçmiş bilgi sınırlıdır. Graf tabanlı bir hafıza sistemi bu problemi çözer. Uzun dönem bellek (long-term memory), kullanıcı etkileşimlerinin, olayların ve ilişkilerin graf yapısı içinde saklanmasıyla oluşturulur. Bu sayede sistem geçmiş bilgileri kalıcı olarak tutabilir.

Kurumsal bağlam (enterprise context) açısından bu yaklaşım son derece kritiktir. Bir organizasyonun geçmiş kararları, projeleri, müşteri etkileşimleri ve süreçleri graf içinde tutulduğunda, yapay zekâ sistemleri yalnızca anlık değil, tarihsel bağlamla da karar verebilir. İlişkisel bellek modeli, bilginin tekil kayıtlar yerine bağlantılar üzerinden hatırlanmasını sağlar. Bu, insan belleğine daha yakın bir modeldir çünkü insanlar da bilgiyi ilişkiler üzerinden hatırlar.

Yapay Zekâ Ajanları ve Graf Üzerinde Gezinme

Yapay zekâ ajanları, belirli bir hedefe ulaşmak için plan yapan ve araç kullanan sistemlerdir. Bu ajanlar graf yapılarıyla birleştiğinde çok daha güçlü hale gelir. Planlama sürecinde ajan, hedefe ulaşmak için hangi düğümler ve ilişkiler üzerinden ilerlemesi gerektiğini belirler. Bu süreç aslında graf üzerinde bir yol bulma problemidir.

Araç kullanımı, ajanların Neo4j gibi dış sistemlerle etkileşime girmesini sağlar. Ajan, graf üzerinde sorgular çalıştırarak bilgi toplar ve bu bilgiyi karar mekanizmasında kullanır.

Multi-hop reasoning, yani çok adımlı akıl yürütme, graf yapılarının en güçlü kullanım alanlarından biridir. Model, bir düğümden diğerine geçerek zincirleme ilişkileri analiz edebilir. Bu, özellikle karmaşık bağımlılık yapılarında kritik öneme sahiptir.

Nedensel zincirler (causal chains) ise olayların neden-sonuç ilişkileri üzerinden analiz edilmesini sağlar. Örneğin bir tedarik zincirinde yaşanan bir kesintinin hangi ürünleri, hangi müşterileri ve hangi bölgeleri etkilediği graf üzerinde bir gezinme ile açık şekilde görülebilir.

Sonuç olarak Neo4j ile yapay zekâ entegrasyonu, yalnızca bir teknolojik birleşim değildir; aynı zamanda yeni bir bilişsel mimaridir. LLM'lerin üretim gücü ile graf veri tabanlarının yapısal doğruluğu birleştiğinde, hem anlam hem de doğruluk açısından çok daha güçlü sistemler ortaya çıkar. Bu yapı, gelecekte yapay zekânın yalnızca metin üreten değil, aynı zamanda ilişkisel dünyayı anlayan sistemlere evrilmesinin temelini oluşturur.

4.11 Sonuç: Veritabanından Anlam Motoruna

Neo4j ve genel olarak graf veri tabanı yaklaşımı, klasik veritabanı sistemlerinden yalnızca teknik bir farklılık taşımaz; aynı zamanda veriyle kurulan düşünme biçimini de değiştirir. Bu nedenle Neo4j'yi sadece bir DBMS (Database Management System) olarak görmek eksik kalır. Çünkü Neo4j'nin sunduğu şey, veriyi saklayan bir sistemden çok, veriyi ilişkilendiren ve bu ilişkiler üzerinden anlam üreten bir yapıdır.

Neo4j'nin klasik DBMS'ten farkı

Klasik ilişkisel veritabanları satır ve sütunlardan oluşan tablolar üzerinden çalışır. Bu yapıda veri merkezlidir; ilişkiler ise çoğu zaman JOIN işlemleriyle sonradan hesaplanır. Yani ilişki, veri modelinin doğal bir parçası değil, sorgu anında türetilen bir sonuçtur.

Neo4j ise bu yaklaşımı tersine çevirir. Burada ilişkiler, veri modelinin en temel bileşenidir. Düğümler kadar ilişkiler de birinci sınıf veri nesnesidir. Bu nedenle sistem, "veriyi nasıl bağlarım?" sorusundan çok "hangi bağlantı zaten var?" sorusuna odaklanır. Bu fark, performansın ötesinde bir düşünce farkıdır.

Veri → ilişki → bağlam dönüşümü

Graf veri modelinin en önemli katkılarından biri, veri dönüşümünü üç aşamalı bir yapıya oturtmasıdır:

1. Veri (Data): Tekil varlıklar ve kayıtlar
2. İlişki (Relation): Bu varlıklar arasındaki bağlantılar
3. Bağlam (Context): İlişkilerin oluşturduğu anlam ağı

Klasik sistemler genellikle ilk aşamada kalır. Yani veri saklanır ve gerektiğinde sorgulanır. Graf yaklaşımı ise ikinci ve üçüncü aşamayı merkeze alır. Özellikle bağlam katmanı, verinin gerçek değerini ortaya çıkarır.

Örneğin bir "ürün" tek başına bir veri parçasıdır. Ancak bu ürünün hangi bileşenlerden oluştuğu, hangi fabrikada üretildiği ve hangi şehirlere gönderildiği bilgisi eklendiğinde artık bir bağlam oluşur. Neo4j tam olarak bu bağlamı görünür hale getirir.

Graf düşüncesinin epistemolojik önemi

Graf yaklaşımı yalnızca bir veri modeli değil, aynı zamanda bir bilgi felsefesidir. Epistemolojik açıdan bakıldığında, geleneksel veri modelleri dünyayı "nesneler" üzerinden anlamaya çalışır. Graf modeli ise dünyayı "ilişkiler ağı" olarak görür.

Bu dönüşüm önemlidir çünkü gerçek dünya nadiren izole nesnelerden oluşur. Aksine her varlık, diğer varlıklarla olan ilişkileri üzerinden anlam kazanır. Bir tedarik zinciri, bir sosyal ağ veya bir finans sistemi tekil noktalardan değil, bu noktalar arasındaki etkileşimlerden oluşur. Graf düşüncesi bu nedenle "varlık merkezli" bir yaklaşımdan "ilişki merkezli" bir yaklaşıma geçişi

temsil eder. Bu geçiş, yalnızca teknik bir değişim değil, aynı zamanda bilişsel bir paradigma değişimidir.

“Veriden Anlama” paradigması

Neo4j ve graf sistemlerinin en önemli katkılarından biri, “Veriden Anlama” paradigmasını mümkün kılmasıdır. Bu paradigma üç temel dönüşüm üzerine kuruludur:

- Veri sadece saklanmaz, ilişkilendirilir
- İlişkiler sadece gösterilmez, analiz edilir
- Analiz edilen ilişkiler anlam üretir

Bu yaklaşımda veri artık pasif bir kaynak değildir. Aksine, sürekli olarak anlam üreten dinamik bir yapıya dönüşür. Özellikle karmaşık sistemlerde bu dönüşüm kritik hale gelir çünkü anlam, tekil verilerden değil, veri noktaları arasındaki etkileşimlerden doğar.

BÖLÜM 5 — BİLGİ GRAFI VE YAPAY ZEKÂ

Bu bölümde amaç, yapay zekânın yalnızca büyük veri üzerinde örüntü öğrenen bir sistem olmaktan çıkıp, “bilgi” ile çalışan, yani anlamlı ilişkiler kurabilen bir yapıya nasıl dönüştüğünü açıklamaktır. Günümüz yapay zekâ sistemleri özellikle büyük dil modelleri sayesinde etkileyici metin üretme yeteneğine ulaşmıştır; ancak bu sistemlerin temel sınırlaması, gerçek anlamda kalıcı ve yapılandırılmış bilgiye dayanmamalarıdır. Bu nedenle bilgi grafı, yapay zekânın eksik kalan “düzenli hafıza” katmanını tamamlayan kritik bir yapı olarak öne çıkar.

Bu bölümde ayrıca, yapay zekâ ile bilgi grafı arasındaki birleşim noktaları incelenecektir. Büyük dil modellerinin bilgi grafı ile nasıl birlikte çalışabileceği, RAG ve GrafRAG gibi yeni mimarilerin neden ortaya çıktığı ve bu yapıların uydurma bilgi üretme sorununu nasıl azaltabileceği ele alınacaktır. Bunun yanında açıklanabilirlik, yani bir yapay zekâ sisteminin neden o sonuca ulaştığını gösterebilme meselesi, bilgi grafı ile doğrudan ilişkilendirilecektir.

Ayrıca tekli ve çoklu ajan sistemlerinin, bilgi grafı üzerinde nasıl bir işbirliği mantığı geliştirdiği, sinirsel ve sembolik yaklaşımların nasıl birleştiği ve graf sinir ağlarının bu yapıya nasıl matematiksel bir temel sunduğu açıklanacaktır. Son olarak, vektör veri tabanları ile graf veri tabanlarının birbirine rakip değil, aslında tamamlayıcı yapılar olduğu gösterilecek ve yapay zekânın uzun dönemli belleği olarak bilgi grafının neden kritik olduğu tartışılacaktır.

5.1 Yapay Zekâ Neden Bilgiye İhtiyaç Duyar?

Yapay zekâ sistemleri uzun süre boyunca “veriden öğrenen” yapılar olarak tasarlandı. Bu yaklaşım özellikle makine öğrenmesi ve derin öğrenme modellerinde oldukça başarılı sonuçlar verdi. Ancak burada temel bir ayrım vardır: veri ile bilgi aynı şey değildir. Veri hamdır, bağlamsızdır ve tek başına anlam taşımaz. Bilgi ise bu verinin düzenlenmiş, ilişkilerle bağlanmış ve yorumlanabilir hale gelmiş biçimidir. Yapay zekânın gerçek anlamda akıllı davranabilmesi için yalnızca veri örüntülerini değil, bu örüntülerin temsil ettiği bilgiyi de kullanabilmesi gerekir.

Günümüz büyük dil modelleri (LLM), internetten ve büyük metin korpuslarından öğrenerek dil üretme kapasitesi kazanmıştır. Ancak bu modellerin temel çalışma prensibi “olasılık tahmini”dir. Yani bir kelimeden sonra hangi kelimenin gelme olasılığının yüksek olduğunu

hesaplarlar. Bu yaklaşım güçlüdür, fakat doğası gereği “anlama” içermez. Model, bir kavramın ne olduğunu bilmez; sadece o kavramın hangi bağlamlarda geçtiğini istatistiksel olarak öğrenir.

İşte bu noktada bilgi grafı devreye girer. Bilgi grafı, varlıklar (kişiler, nesneler, olaylar) ile bu varlıklar arasındaki ilişkileri açık biçimde temsil eder. Bu yapı, yapay zekâya yalnızca kelime dizileri değil, yapılandırılmış bir dünya modeli sunar. Örneğin “Einstein → geliştirdi → Görelilik Teorisi” gibi bir ilişki, modelin yalnızca metin içinde değil, açık bir bilgi yapısı içinde de bu bağlantıyı görmesini sağlar.

Yapay zekânın bilgiye ihtiyaç duymasının bir diğer nedeni de tutarlılıktır. Saf dil modelleri zaman zaman çelişkili veya uydurma bilgiler üretebilir. Çünkü bu sistemler doğruluk kontrolü yapmaz, yalnızca olasılık üretir. Bilgi grafı ise doğrulanmış ilişkiler içerdiği için bir tür “gerçeklik iskeleti” görevi görür. Bu iskelet, yapay zekânın daha güvenilir ve açıklanabilir sonuçlar üretmesine yardımcı olur.

Ayrıca bilgi, yapay zekânın genelleme kapasitesini de artırır. Sadece veri üzerinden öğrenen bir model, yeni bir durumla karşılaştığında zorlanabilir. Ancak ilişkisel bilgi yapıları sayesinde model, daha önce görmediği durumları mevcut ilişkiler üzerinden yorumlayabilir. Bu, insan düşüncesine daha yakın bir öğrenme biçimidir.

Sonuç olarak yapay zekâ yalnızca veriyle değil, bilgiyle beslendiğinde daha güçlü, daha tutarlı ve daha açıklanabilir hale gelir. Bilgi grafı, bu dönüşümün en kritik araçlarından biridir çünkü yapay zekâya yalnızca “ne olduğunu” değil, “nasıl ilişkili olduğunu” da gösterir.

5.2 Büyük Dil Modelleri (LLM) ve Bilgi Grafı Birleştirme

Büyük dil modelleri, doğal dil işleme alanında devrim yaratmıştır. Bu modeller, milyarlarca parametre üzerinden dil örüntülerini öğrenerek insan benzeri metinler üretebilir. Ancak bu modellerin temel sınırlaması, bilgiye doğrudan erişim yerine istatistiksel örüntülere dayanmasıdır. Bu durum, özellikle doğruluk, güncellik ve tutarlılık gerektiren uygulamalarda önemli sorunlar yaratır. İşte bu noktada bilgi grafı ile LLM’lerin birleştirilmesi, yapay zekâ sistemlerinde yeni bir paradigma ortaya çıkarmıştır.

Bilgi grafı, dünyadaki varlıkları ve bu varlıklar arasındaki ilişkileri yapılandırılmış bir biçimde temsil eder. LLM ise bu yapıların üzerinde doğal dil üretme yeteneği sağlar. Bu iki yaklaşım birleştirildiğinde, hem “bilgi tabanı” hem de “dil üretim motoru” birlikte çalışır. Böylece sistem hem neyi bildiğini hem de bunu nasıl ifade edeceğini daha kontrollü şekilde yönetebilir.

Bu birleşimin en önemli kullanım alanlarından biri bilgi getirmeli üretim (Retrieval-Augmented Generation, RAG) mimarileridir. Bu yapıda LLM, cevap üretmeden önce dış bir bilgi kaynağından veri çeker. Bilgi grafı burada güçlü bir kaynak olarak kullanılabilir çünkü ilişkisel ve doğrulanmış bilgi sağlar. Bu sayede model, yalnızca kendi ağırlıklarına dayanmak yerine dış dünyaya referans vererek yanıt üretir.

Bilgi grafı ile LLM entegrasyonunun bir diğer avantajı açıklanabilirliktir. Geleneksel LLM’ler “kara kutu” olarak çalışır; yani neden belirli bir cevabı ürettikleri kolayca açıklanamaz. Ancak

bilgi grafı kullanıldığında, modelin hangi düğümler ve ilişkiler üzerinden sonuca ulaştığı takip edilebilir. Bu durum özellikle sağlık, finans ve hukuk gibi kritik alanlarda büyük önem taşır.

Ayrıca bilgi grafı, LLM'lerin "uydurma bilgi üretme" (hallucination) problemini azaltır. Model, graf içinde var olmayan bir ilişkiyi üretmek yerine mevcut ilişkiler üzerinden yanıt oluşturur. Bu, sistemin güvenilirliğini önemli ölçüde artırır.

Teknik açıdan bakıldığında, bu entegrasyon genellikle üç katmanda gerçekleşir: (1) bilgi grafından ilgili alt ağın çıkarılması, (2) bu bilginin metin formatına dönüştürülmesi, (3) LLM'in bu bağlamı kullanarak yanıt üretmesi. Bu yapı, hem sembolik hem de istatistiksel yapay zekâ yaklaşımlarını bir araya getirir.

Sonuç olarak LLM ve bilgi grafı birleşimi, yapay zekânın yalnızca metin üreten bir sistem olmaktan çıkıp, bilgiyle çalışan ve bu bilgiyi yapılandırarak kullanan bir sisteme dönüşmesini sağlar. Bu yaklaşım, geleceğin yapay zekâ mimarilerinin temel taşlarından biri olarak görülmektedir.

5.3 Bilgi Getirmeli Üretim (Retrieval-Augmented Generation, RAG)

Bilgi Getirmeli Üretim, kısa adıyla RAG (Retrieval-Augmented Generation), büyük dil modellerinin (LLM) en önemli sınırlamalarından birini aşmak için geliştirilmiş hibrit bir yapay zekâ mimarisidir. Bu sınırlama, modelin eğitim sırasında gördüğü bilgilerle sınırlı kalması ve güncel, doğrulanabilir dış bilgiye doğrudan erişememesidir. RAG yaklaşımı, bu problemi çözmek için üretim sürecini iki aşamaya ayırır: Bilgi getirme (retrieval) ve metin üretme (generation).

Bu mimaride ilk aşama, kullanıcının sorusuna en uygun bilgilerin dış bir veri kaynağından çekilmesidir. Bu kaynak bir belge arşivi, veri tabanı, web koleksiyonu veya vektör tabanı olabilir. Sistem, soruyu analiz eder ve anlamına en yakın belgeleri veya bilgi parçalarını bulur. Bu aşama, klasik arama motorlarının yaptığı işlemin daha gelişmiş ve anlam tabanlı bir versiyonudur. Ana fark, anahtar kelime eşleşmesi yerine anlamsal benzerliklerin kullanılmasıdır.

İkinci aşama ise büyük dil modelinin devreye girdiği üretim aşamasıdır. Burada model, dışarıdan getirilen bu bilgileri bağlam olarak kullanır ve doğal dilde bir yanıt üretir. Böylece model yalnızca kendi iç parametrelerine dayanarak değil, aynı zamanda dış dünyadan gelen güncel ve doğrulanmış bilgilerle birlikte çalışır. Bu durum, yapay zekânın "kapalı sistem" olmaktan çıkıp "bilgiye açık sistem" haline gelmesini sağlar.

RAG yaklaşımının en önemli avantajlarından biri doğruluk oranını artırmasıdır. Geleneksel büyük dil modelleri, bazen ikna edici ama yanlış bilgiler üretebilir. Çünkü bu modeller gerçeklik kontrolü yapmaz; yalnızca olasılıksal dil üretimi gerçekleştirir. RAG sistemlerinde ise cevap, dış bir bilgi kaynağına dayandığı için daha güvenilir hale gelir. Bu, özellikle akademik araştırma, tıbbi danışmanlık ve teknik bilgi gerektiren alanlarda kritik bir fark yaratır.

Bir diğer önemli avantaj güncelliktir. Büyük dil modelleri belirli bir eğitim kesitine sahiptir ve bu kesitin ötesindeki gelişmeleri bilemezler. RAG ise sürekli güncellenen veri kaynaklarıyla

çalışabildiği için modelin bilgi tabanı dinamik hale gelir. Bu, sistemi statik bir model olmaktan çıkarıp yaşayan bir bilgi sistemi haline getirir.

RAG aynı zamanda maliyet açısından da avantajlıdır. Modelin tamamını yeniden eğitmek yerine yalnızca bilgi kaynağı güncellenir. Bu da hem zaman hem de hesaplama maliyetlerini önemli ölçüde düşürür.

Ancak RAG mimarisinin bazı sınırlamaları da vardır. Özellikle bilgi getirme aşamasının kalitesi, tüm sistemin performansını doğrudan etkiler. Eğer yanlış veya alakasız belgeler getirilirse, üretim aşaması da hatalı sonuçlar verebilir. Bu nedenle RAG sistemlerinde bilgi indeksleme ve arama algoritmaları kritik öneme sahiptir.

Sonuç olarak RAG, yapay zekâ sistemlerini daha güvenilir, daha güncel ve daha açıklanabilir hale getiren temel bir mimaridir. Bu yaklaşım, büyük dil modellerini kapalı bilgi sistemlerinden çıkararak, dış dünyaya bağlı dinamik bilgi sistemlerine dönüştürür. Bu dönüşüm, modern yapay zekânın en önemli kırılma noktalarından biridir.

5.4 Graf Tabanlı RAG Yaklaşımı (GrafRAG)

Graf Tabanlı RAG, yani GrafRAG, klasik bilgi getirmeli üretim (RAG) yaklaşımının daha gelişmiş ve yapısal bir versiyonudur. Standart RAG sistemlerinde bilgi genellikle belge parçaları veya metin kesitleri üzerinden getirilirken, GrafRAG yaklaşımında bilgi graf yapısı üzerinden organize edilir ve sorgulanır. Bu durum, yalnızca metin benzerliğine dayalı aramadan, ilişki temelli ve bağlamsal bir bilgi keşfine geçiş anlamına gelir.

GrafRAG'in temel fikri şudur: bilgi yalnızca metin içinde değil, aynı zamanda varlıklar (kişiler, kavramlar, olaylar) ve bu varlıklar arasındaki ilişkiler içinde de saklanır. Bu nedenle sistem, bir soruyu yanıtlamak için yalnızca benzer metinleri bulmaz; aynı zamanda ilgili düğümler ve bağlantılar üzerinden bir alt bilgi ağı oluşturur. Bu alt ağ, büyük grafın içinden çıkarılan anlamlı bir "bilgi alt evreni" gibidir.

Örneğin bir kullanıcı "Einstein'ın bilimsel etkileri nelerdir?" gibi bir soru sorduğunda, klasik RAG sistemi bu soruya benzer metin parçalarını getirir. Ancak GrafRAG sistemi, Einstein düğümünden başlayarak onun bağlantılı olduğu teorileri, öğrencileri, çalışmaları ve etkileşimde bulunduğu diğer bilim insanlarını içeren bir alt graf oluşturur. Bu yapı, çok daha zengin ve bağlamsal bir bilgi sunar.

GrafRAG'in en önemli avantajı bağlamsal derinliktir. Çünkü bilgi yalnızca tekil parçalar halinde değil, ilişkiler ağı içinde sunulur. Bu sayede model, daha tutarlı ve bütüncül cevaplar üretebilir. Özellikle karmaşık sorular, çok adımlı ilişkiler ve neden-sonuç analizleri gerektiren durumlarda GrafRAG klasik RAG'e göre daha güçlü performans gösterir.

Bir diğer önemli avantaj, bilgi keşfi yeteneğidir. GrafRAG yalnızca doğrudan ilgili bilgileri değil, dolaylı ilişkileri de ortaya çıkarabilir. Bu, graf yapısının doğal bir özelliğidir. Örneğin iki varlık arasında doğrudan bir bağlantı olmasa bile, ortak bir düğüm üzerinden ilişki kurulabilir. Bu durum, yeni bilgi keşiflerine ve beklenmeyen bağlantıların ortaya çıkmasına olanak tanır.

GrafRAG aynı zamanda açıklanabilirlik açısından da güçlüdür. Çünkü sistem hangi düğümler ve ilişkiler üzerinden sonuca ulaştığını açıkça gösterebilir. Bu, özellikle kritik karar destek sistemlerinde büyük bir avantaj sağlar. Kullanıcı yalnızca sonucu değil, bu sonuca giden bilgi yolunu da görebilir.

Ancak GrafRAG sistemlerinin kurulumu ve yönetimi klasik RAG'e göre daha karmaşıktır. Çünkü öncelikle iyi tasarlanmış bir bilgi grafı gerekir. Bu grafın sürekli güncellenmesi, varlıkların doğru şekilde eşleştirilmesi ve ilişkilerin doğru modellenmesi gerekir. Bu nedenle GrafRAG, daha çok olgun veri altyapısına sahip sistemlerde kullanılır.

Sonuç olarak GrafRAG, yapay zekâda bilgi temelli üretimi bir adım daha ileri taşıyan bir yaklaşımdır. Klasik RAG metin parçaları üzerinden çalışırken, GrafRAG ilişkiler üzerinden çalışır. Bu fark, yapay zekânın yalnızca metni anlamasını değil, aynı zamanda "dünyanın yapısını" anlamasını sağlar. Bu nedenle GrafRAG, geleceğin bilgi tabanlı yapay zekâ mimarilerinin en kritik bileşenlerinden biri olarak görülmektedir.

5.5 Uydurma Bilgi Üretme (Hallucination) Problemi

Uydurma bilgi üretme problemi, büyük dil modellerinin (LLM) en kritik ve en çok tartışılan sınırlamalarından biridir. Bu problem, modelin gerçekte var olmayan bilgileri, varmış gibi son derece ikna edici bir dil ile üretmesi durumunu ifade eder. Buradaki temel sorun, modelin "doğruluk" kavramına sahip olmamasıdır. LLM'ler, doğruyu yanlıştan ayıran bir iç denetim mekanizmasına sahip değildir; yalnızca dilsel örüntüler üzerinden en olası kelime dizisini üretirler. Bu nedenle ortaya çıkan metin, dil açısından tutarlı olsa bile bilgi açısından hatalı olabilir.

Bu durumun temel nedeni, büyük dil modellerinin çalışma prensibidir. Bu modeller, devasa metin koleksiyonları üzerinde eğitilir ve kelimeler arasındaki istatistiksel ilişkileri öğrenir. Ancak bu öğrenme süreci, dış dünyada gerçekten neyin doğru olduğuna dair bir "zemin" içermez. Model, "Einstein fizikçidir" ifadesini çok sayıda bağlamda gördüğü için doğru kabul eder; ancak "Einstein 2025 yılında kuantum bilgisayar geliştirdi" gibi bir ifadeyi de benzer dil kalıplarına uyuyorsa üretebilir. Bu durumda model, gerçeklikten kopuk ama dilsel olarak inandırıcı bir içerik üretmiş olur.

Uydurma bilgi üretme problemi özellikle üç durumda daha belirgin hale gelir. Birincisi, modelin eğitim verisinde yeterli bilgi bulunmayan konulardır. Bu durumda model boşluğu doldurmak için genelleme yapar ve çoğu zaman yanlış sonuçlar üretir. İkincisi, çok spesifik ve güncel bilgi gerektiren sorulardır. Modelin eğitim kesiti belirli bir tarihte durduğu için, güncel olaylar hakkında tahmin üretir. Üçüncüsü ise çok adımlı mantık gerektiren sorulardır; model bu tür durumlarda ara adımları yanlış kurabilir.

Bu problemin en tehlikeli yönü, üretilen yanlış bilginin son derece ikna edici olmasıdır. Dil modeli, akademik bir tonla, referanslı gibi görünen ama gerçekte var olmayan bilgiler üretebilir. Bu durum, özellikle sağlık, hukuk ve finans gibi kritik alanlarda ciddi riskler oluşturur.

Uydurma bilgi üretme problemi yalnızca teknik bir hata değil, aynı zamanda epistemolojik bir sorundur. Çünkü burada “bilgi nedir?” sorusu yeniden gündeme gelir. Eğer bir sistem doğruyu ayırt edemiyor ama doğruya benzeyen ifadeler üretebiliyorsa, bu üretim ne kadar güvenilirdir? Bu soru, yapay zekâ çağının en temel felsefi tartışmalarından biridir.

Bu problemi azaltmak için çeşitli yöntemler geliştirilmiştir. Bunların başında bilgi getirmeli üretim (RAG) ve graf tabanlı bilgi sistemleri gelir. Bu sistemlerde model yalnızca kendi iç bilgisini kullanmaz; dış kaynaklardan doğrulanmış bilgi alır. Böylece üretim süreci daha kontrollü hale gelir. Ayrıca bilgi grafı gibi yapılandırılmış sistemler, modelin yanlış ilişki kurmasını engelleyerek daha güvenilir sonuçlar üretmesini sağlar.

Sonuç olarak uydurma bilgi üretme problemi, büyük dil modellerinin en önemli sınırlamalarından biridir ve bu sınır, yapay zekânın “dil üretme” ile “bilgi üretme” arasındaki farkı anlamasını zorunlu kılar. Gelecekte bu problemin çözümü, yalnızca daha büyük modellerle değil, daha iyi bilgi yapılarıyla mümkün olacaktır.

5.6 Açıklanabilir Yapay Zekâ (Explainable YZ) ve Anlamsal İzlenebilirlik

Açıklanabilir yapay zekâ (Explainable YZ, XAI), bir yapay zekâ sisteminin verdiği kararların insanlar tarafından anlaşılabilir ve izlenebilir olmasını amaçlayan bir yaklaşımdır. Geleneksel derin öğrenme modelleri genellikle “kara kutu” olarak çalışır. Yani bir giriş verisi verildiğinde bir çıktı üretirler, ancak bu çıktının nasıl üretildiği çoğu zaman açık değildir. Bu durum, özellikle kritik kararların alındığı alanlarda önemli bir güven problemi yaratır.

Açıklanabilirlik ihtiyacı, yapay zekânın günlük hayata daha fazla entegre olmasıyla birlikte daha da önem kazanmıştır. Bir sistemin yalnızca doğru sonuç üretmesi yeterli değildir; aynı zamanda bu sonuca nasıl ulaştığını da açıklayabilmesi gerekir. Örneğin bir sağlık sisteminin “bu hastada risk yüksektir” demesi yeterli değildir; bu sonuca hangi belirtiler, hangi veriler ve hangi ilişkiler üzerinden ulaşıldığının da gösterilmesi gerekir.

Bu noktada bilgi grafı, açıklanabilir yapay zekâ için son derece güçlü bir araç haline gelir. Çünkü bilgi grafı, veriyi düğümler ve ilişkiler şeklinde açık bir yapıda saklar. Bu yapı sayesinde bir yapay zekâ sisteminin karar süreci geriye doğru izlenebilir. Yani modelin hangi düğümlerden geçtiği, hangi ilişkileri kullandığı ve hangi bilgi yolunu izlediği açıkça görülebilir.

Anlamsal izlenebilirlik (semantic traceability), bu sürecin daha ileri bir aşamasıdır. Bu kavram, yalnızca teknik karar adımlarını değil, aynı zamanda anlam düzeyindeki bağlantıları da izlemeyi ifade eder. Örneğin bir model bir öneri sunduğunda, bu önerinin hangi kavramsal ilişkiler üzerinden üretildiği de takip edilebilir. Bu, yalnızca “nasıl” sorusunu değil, “neden” sorusunu da yanıtlamayı mümkün kılar.

Açıklanabilir yapay zekâ, özellikle düzenleyici kurumlar açısından da büyük önem taşır. Avrupa Birliği’nin veri düzenlemeleri ve benzeri yasal çerçeveler, yapay zekâ sistemlerinin karar süreçlerinin şeffaf olmasını giderek daha fazla zorunlu hale getirmektedir. Bu nedenle XAI, yalnızca teknik bir özellik değil, aynı zamanda yasal bir gereklilik haline gelmektedir.

Bilgi grafı ile desteklenen açıklanabilirlik sistemlerinde, her karar bir “yol” olarak temsil edilebilir. Bu yol, başlangıç verisinden nihai karara kadar olan tüm ilişkisel adımları içerir. Kullanıcı bu yolu inceleyerek sistemin nasıl düşündüğünü anlayabilir. Bu, yapay zekâ ile insan arasında güven ilişkisini güçlendirir.

Anlamsal izlenebilirlik aynı zamanda hata analizini de kolaylaştırır. Eğer sistem yanlış bir karar üretirse, bu kararın hangi ilişki veya veri düğümünden kaynaklandığı tespit edilebilir. Böylece sistem sadece sonuç düzeyinde değil, süreç düzeyinde de debug edilebilir hale gelir.

Sonuç olarak açıklanabilir yapay zekâ ve anlamsal izlenebilirlik, yapay zekânın yalnızca güçlü değil, aynı zamanda güvenilir ve denetlenebilir olmasını sağlar. Bilgi grafı ile birleştiğinde, yapay zekâ sistemleri ilk kez hem karar verebilen hem de bu kararları anlamlı şekilde açıklayabilen yapılar haline gelir. Bu da yapay zekânın toplumsal kabulü açısından kritik bir dönüşümdür.

5.7 Bilgi Grafı Destekli Özerk Sistemler (Ajan Sistemleri)

Özerk sistemler, yani “ajanlar”, belirli bir hedef doğrultusunda çevresini algılayan, karar veren ve eylem gerçekleştiren yapay zekâ bileşenleridir. Bu sistemlerin en önemli özelliği, yalnızca tek seferlik çıktı üretmek yerine bir süreç boyunca kendi davranışını düzenleyebilmesidir. Ancak modern özerk sistemlerin karşılaştığı en temel sorunlardan biri, uzun süreli hafıza eksikliği ve bağlamsal tutarsızlıktır. İşte bu noktada bilgi grafı, ajan sistemlerine yapısal bir bellek ve anlamlı bir dünya modeli kazandırır.

Bilgi grafı destekli özerk sistemlerde, ajanın bilgisi yalnızca metin veya parametre ağırlıkları içinde değil, açıkça tanımlanmış düğümler (varlıklar) ve ilişkiler (bağlantılar) içinde saklanır. Bu yapı sayesinde ajan, “ne biliyorum?” sorusuna yalnızca olasılıksal bir yanıt vermek yerine, doğrudan erişilebilir bir bilgi haritası üzerinden cevap verebilir. Örneğin bir araştırma ajanı, bir akademisyenin geçmiş çalışmalarını, ilişkili makalelerini ve etkileşimde bulunduğu diğer araştırmacıları bilgi grafı üzerinden takip edebilir.

Bu yaklaşımın en önemli katkısı, ajanın karar verme sürecine açıklık ve izlenebilirlik kazandırmasıdır. Klasik büyük dil modeli tabanlı ajanlar, kararlarını içsel ağırlıklar üzerinden verdiği için “neden bu kararı verdi?” sorusu çoğu zaman net şekilde yanıtlanamaz. Ancak bilgi grafı kullanıldığında, ajanın izlediği yol bir tür “ilişkisel düşünme zinciri” olarak açıkça görülebilir. Bu zincir, hangi düğümlerden geçtiğini ve hangi ilişkileri kullandığını gösterir.

Bilgi grafı destekli ajanlar özellikle uzun vadeli görevlerde büyük avantaj sağlar. Örneğin bir proje yönetim ajanı, görevleri, ekip üyelerini, kaynakları ve zaman planını bir graf yapısı içinde temsil ederek daha tutarlı kararlar verebilir. Aynı şekilde bir akademik araştırma ajanı, literatür taramasını yalnızca anahtar kelimelerle değil, kavramsal ilişkiler üzerinden gerçekleştirebilir.

Bu sistemlerin bir diğer önemli avantajı bağlam sürekliliğidir. Klasik sistemlerde her etkileşim büyük ölçüde bağımsızdır. Ancak bilgi grafı kullanıldığında, önceki etkileşimler graf üzerinde kalıcı yapılar oluşturur. Böylece ajan, geçmiş deneyimlerinden öğrenerek daha tutarlı davranışlar sergiler.

Ayrıca bilgi grafı, ajanların “hatırlama” kapasitesini yapısal hale getirir. İnsan hafızası nasıl ilişkiler üzerinden çalışıyorsa, bu sistemler de benzer şekilde varlıklar arası bağlantılar üzerinden bilgiye erişir. Bu, yapay zekânın insan benzeri bilişsel yapıya yaklaşmasını sağlar.

Sonuç olarak bilgi grafı destekli özerk sistemler, yapay zekânın yalnızca yanıt veren bir araç olmaktan çıkıp, bilgiye dayalı karar veren bir “dijital aktör” haline gelmesini sağlar. Bu yapı, özellikle karmaşık ve uzun vadeli problem çözme süreçlerinde yeni bir mimari standart oluşturmaktadır.

5.8 Çoklu Ajan (Multi-Agent) Yapıları

Çoklu ajan sistemleri (multi-agent systems), birden fazla özerk yapay zekâ ajanın birlikte çalışarak karmaşık problemleri çözmesini ifade eder. Bu yaklaşım, tek bir büyük modelin tüm problemi çözmeye çalışması yerine, problemi küçük parçalara bölen ve her parçayı farklı uzman ajanlara dağıtan bir mimariyi temel alır. Bu yapı, özellikle dağıtık zekâ (distributed intelligence) fikrinin yapay zekâ dünyasındaki karşılığıdır.

Çoklu ajan sistemlerinin temel avantajı, uzmanlaşma ve iş bölümü yeteneğidir. Her ajan belirli bir görev üzerinde uzmanlaşabilir: biri veri toplama, diğeri analiz, bir diğeri karar verme süreçlerinden sorumlu olabilir. Bu yapı, insan organizasyonlarına benzer şekilde çalışır. Tıpkı bir ekip içinde farklı uzmanların farklı görevleri üstlenmesi gibi, ajanlar da sistem içinde rol paylaşımı yapar.

Bilgi grafı, çoklu ajan sistemlerinde merkezi bir koordinasyon katmanı olarak görev yapar. Ajanlar arasındaki iletişim doğrudan değil, bilgi grafı üzerinden gerçekleşebilir. Bu sayede sistemde tutarlılık sağlanır ve bilgi çatışmaları azaltılır. Her ajan, graf üzerinde güncellemeler yaparak ortak bilgi havuzuna katkıda bulunur.

Çoklu ajan sistemlerinde en önemli konulardan biri koordinasyondur. Ajanlar bağımsız hareket ettiklerinde çakışan kararlar veya tutarsız sonuçlar üretebilirler. Bu problemi çözmek için bilgi grafı, ortak bir “gerçeklik modeli” sağlar. Böylece tüm ajanlar aynı bilgi yapısı üzerinden çalışır.

Bir diğer önemli konu iletişim protokolleridir. Ajanlar birbirleriyle doğrudan mesajlaşabilir veya dolaylı olarak bilgi grafı üzerinden etkileşime girebilir. Graf tabanlı iletişim, özellikle büyük sistemlerde daha ölçeklenebilir ve düzenli bir yapı sunar. Çünkü bilgi merkezi olarak graf, tüm değişiklikleri kayıt altına alır.

Çoklu ajan sistemleri özellikle karmaşık problem alanlarında güçlüdür. Örneğin finansal piyasa analizi, lojistik optimizasyon, bilimsel keşif süreçleri ve büyük ölçekli veri analitiği gibi alanlarda farklı ajanların birlikte çalışması, tek bir modelin ulaşamayacağı seviyede performans sağlayabilir.

Bu sistemlerin bir diğer önemli özelliği dayanıklılıktır. Tek bir ajan başarısız olduğunda sistem tamamen çökmez; diğer ajanlar görevine devam eder. Bu, sistemin hata toleransını artırır ve daha güvenilir bir yapı oluşturur.

Bilgi grafı ile birleştiğinde çoklu ajan sistemleri, yalnızca dağıtık hesaplama değil, aynı zamanda dağıtık bilgi yönetimi haline gelir. Her ajan hem bilgi üretir hem de graf yapısını günceller. Bu sürekli güncellenen yapı, sistemin zamanla öğrenmesini ve gelişmesini sağlar.

Sonuç olarak çoklu ajan yapıları, yapay zekânın bireysel zekâdan kolektif zekâya geçişini temsil eder. Bilgi grafı bu yapıda hem iletişim omurgası hem de ortak hafıza olarak kritik bir rol oynar. Bu birleşim, geleceğin yapay zekâ sistemlerinin temel mimari taşlarından biri olacaktır.

5.9 Sinirsel-Sembolik Yapay Zekâ (Neuro-Symbolic YZ)

Sinirsel-sembolik yapay zekâ, yapay zekâ alanında uzun süredir birbirine alternatif gibi görünen iki yaklaşımı bir araya getirme girişimidir: sinirsel (nöral) öğrenme ve sembolik bilgi işleme. Sinirsel sistemler, özellikle derin öğrenme modelleri, büyük veri üzerinden örüntü öğrenmede son derece başarılıdır. Ancak bu sistemler çoğu zaman açıklanabilirlikten yoksundur ve öğrenilen bilgi açık bir şekilde temsil edilmez. Buna karşılık sembolik yapay zekâ, kurallara dayalı, mantıksal ve açık bilgi temsiline dayanır; fakat esnek öğrenme ve büyük veriyle baş etme konusunda zayıftır. Sinirsel-sembolik yaklaşım, bu iki dünyanın güçlü yönlerini birleştirmeyi hedefler.

Bu yaklaşımın temel fikri şudur: öğrenme sinirsel ağlar üzerinden gerçekleşir, ancak bu öğrenme sembolik bir yapıya dönüştürülerek temsil edilir ve bu yapı üzerinden akıl yürütme yapılır. İşte burada bilgi grafı kritik bir köprü rolü üstlenir. Çünkü bilgi grafı, sembolik bilginin doğal temsil biçimidir. Varlıklar ve ilişkiler açıkça tanımlandığı için, sinirsel modellerin ürettiği sonuçlar bu yapıya entegre edilebilir.

Sinirsel-sembolik sistemlerin en önemli avantajlarından biri genelleme kapasitesidir. Saf sinirsel modeller belirli veri dağılımlarında iyi çalışırken, sembolik yapı sayesinde daha soyut ve yapılandırılmış genellemeler yapılabilir. Örneğin bir model “tüm memeliler canlıdır” gibi bir kuralı sembolik olarak temsil edebilir ve bu bilgiyi yeni durumlara uygulayabilir. Bu, yalnızca istatistiksel öğrenme ile elde edilmesi zor bir yetenektir.

Bir diğer önemli avantaj açıklanabilirliktir. Sinirsel ağlar genellikle “kara kutu” olarak görülürken, sembolik katman sayesinde karar süreçleri daha şeffaf hale gelir. Modelin hangi kurallara veya hangi ilişkisel yapılara dayanarak karar verdiği izlenebilir. Bu, özellikle kritik alanlarda güvenilirlik açısından büyük önem taşır.

Bilgi grafı burada yalnızca bir veri yapısı değil, aynı zamanda bir “anlam omurgası” olarak görev yapar. Sinirsel model tarafından öğrenilen temsiller, bu omurgaya yerleştirilerek daha düzenli ve kontrol edilebilir hale getirilir. Bu sayede yapay zekâ hem esnek öğrenme kapasitesine hem de yapılandırılmış bilgi temsiline sahip olur.

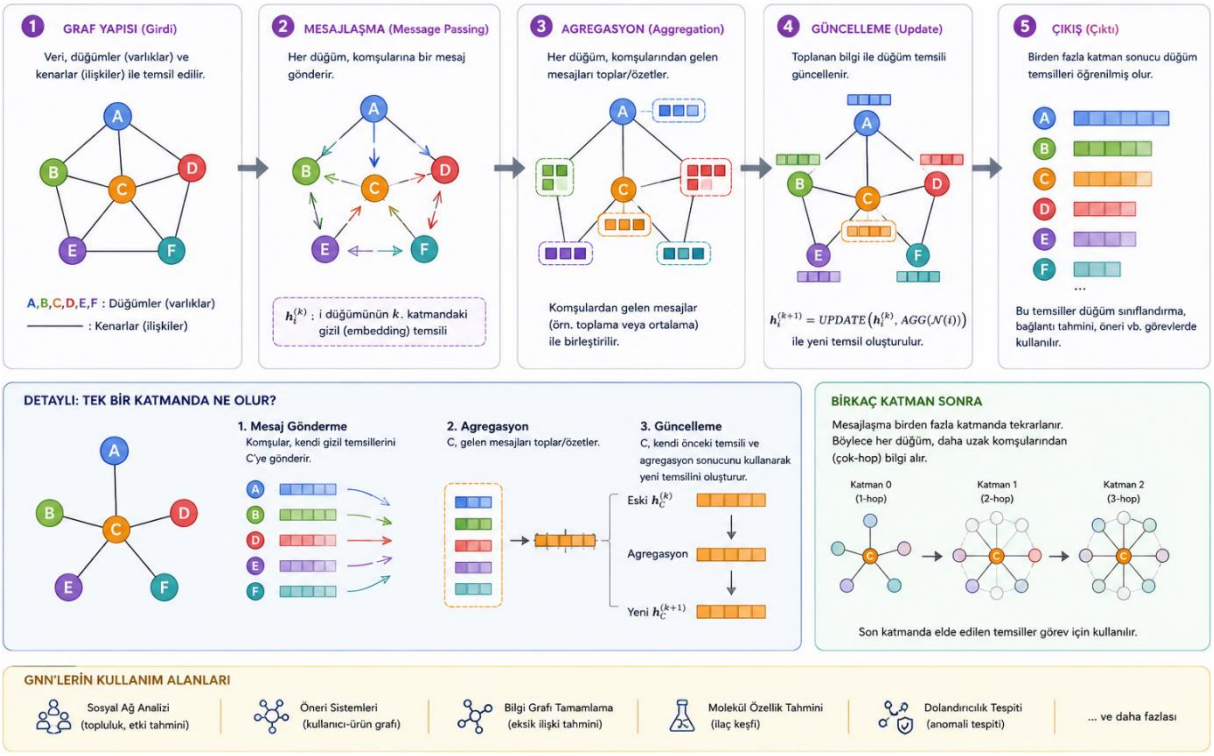
Sinirsel-sembolik yapay zekâ aynı zamanda bilişsel bilimlerle de yakından ilişkilidir. İnsan zihni de yalnızca istatistiksel öğrenme değil, aynı zamanda sembolik düşünme ve kurallar üzerinden akıl yürütme kapasitesine sahiptir. Bu nedenle bu yaklaşım, insan benzeri düşünme sistemlerine daha yakın bir model olarak görülmektedir.

Sonuç olarak sinirsel-sembolik yapay zekâ, yapay zekânın iki temel paradigmasını birleştirerek daha güçlü, daha açıklanabilir ve daha esnek sistemler oluşturmayı hedefler. Bilgi grafi bu birleşimde merkezi bir rol oynar ve yapay zekânın yalnızca öğrenen değil, aynı zamanda "anlam kuran" bir yapıya dönüşmesini sağlar.

5.10 Graf Sinir Ağları (Graf Neural Networks, GNN)

Graf Sinir Ağları (Graf Neural Networks, GNN), doğrudan graf yapıları üzerinde çalışan özel bir yapay zekâ mimarisidir. Geleneksel sinir ağları genellikle görüntü ve zaman serisi gibi düzenli (grid veya sequence tabanlı) veri temsilleri üzerinde çalışırken, metin gibi değişken uzunluklu sıralı verileri de vektör temsillerine dönüştürerek işler. Buna karşılık GNN'ler, düğümler ve bu düğümler arasındaki ilişkilerden oluşan düzensiz (non-Euclidean) graf yapıları üzerinde çalışmak üzere tasarlanmıştır. Bu nedenle bilgi grafi yapılarıyla doğal ve doğrudan bir uyum sergiler.

Şekil 18. Graph Neural Network (GNN) Nasıl Çalışır?



Şekil 18, derin öğrenmenin en heyecan verici alanlarından biri olan Graf Sinir Ağları'nın (Graph Neural Networks - GNN) temel çalışma mekanizmasını adım adım açıklayan, akademik hassasiyeti yüksek ve harika tasarlanmış bir infografiktir.

Şema; üst kısımdaki 5 ana adımdan oluşan doğrusal akış, alt soldaki tek bir katmanın mikro analizi, alt sağdaki katmanların derinleşmesi (hop mekanizması) ve en alttaki kullanım alanları olmak üzere 4 bölümden oluşuyor.

I. ÜST BÖLÜM: GNN'in Temel Akış Adımları (1-5)

Graf Sinir Ağları, geleneksel yapay sinir ağları gibi matrisleri değil, doğrudan ağ yapılarını (graf) işler. Bu işlem döngüsü şu şekilde gerçekleşir:

- **1. GRAF YAPISI (Girdi):** Sürecin ham halidir. Elimizde harflerle adlandırılmış düğümler (A, B, C, D, E, F varlıkları) ve bunları birbirine bağlayan kenarlar (ilişkiler) bulunur. Her düğümün arkasında, onu tanımlayan sayısal birer vektör (özellik) vardır.
- **2. MESAJLAŞMA (Message Passing):** GNN'in kalbidir. Graptaki her düğüm, komşu olduğu diğer düğümlere kendi sahip olduğu bilgiyi bir "mesaj" olarak gönderir. Formül kutusunda ifade edilen terim, *i düğümünün k. katmandaki gizil (embedding) temsilini*, yani o anki sayısal kimliğini gösterir.
- **3. AGREGASYON (Aggregation):** Her düğüm, komşularından gelen bu çeşitli mesajları toplar ve özetler. Görselde C düğümünün etrafındaki renkli kutucuklar, komşuları A, B, D, E ve F'den gelen bilgileri topladığını sembolize eder. Bu birleştirme işlemi genellikle toplama, ortalama alma veya maksimumu seçme gibi permütasyondan etkilenmeyen matematiksel fonksiyonlarla yapılır.
- **4. GÜNCELLEME (Update):** Düğüm, komşularından topladığı bu ortak bilgiyi (agregasyon sonucu) alır ve kendi eski bilgisiyle harmanlayarak kendini günceller. Matematiksel olarak bu durum şu formülle ifade edilmiştir:
- **5. ÇIKIŞ (Çıktı):** Bu döngü katmanlar boyunca tekrar edildikten sonra, her düğüm graf içindeki konumunu ve çevresini yansıtan nihai birer vektör temsiline (embedding) kavuşur. Görselde her düğümün yanında farklı renklerde ve uzunluklarda şeritler (vektörler) görüyorsunuz. Artık bu vektörler; düğüm sınıflandırma, bağlantı tahmini (link prediction) veya öneri sistemleri gibi görevlerde kullanılmaya hazırdır.

II. ALT SOL BÖLÜM: Detaylı - Tek Bir Katmanda Ne Olur?

Bu panel, yukarıdaki adımları merkezdeki C düğümü üzerinden mikro ölçekte simüle eder:

1. **Mesaj Gönderme:** C'nin komşuları (A, B, D, E, F) kendi gizil şeritlerini (vektörlerini) oklarla C'ye doğru fırlatır.
2. **Agregasyon:** Gelen bu 5 farklı renkli şerit, tek bir ortak potada eritilerek ortalama bir vektöre (turuncu/kahverengi şerit) indirgenir.
3. **Güncelleme:** C düğümü, kendi eski turuncu şeridini ve komşulardan gelen bu yeni agregasyon şeridini birleştirerek yepyeni, daha rafine bir vektör üretir.

III. ALT SAĞ BÖLÜM: Birkaç Katman Sonra (Hop Mekanizması)

GNN'lerde katman sayısının ne anlama geldiğini gösteren çok kritik bir bölümdür:

- **Katman 0 (1-hop):** C düğümü sadece doğrudan bağlı olduğu komşularından (1 adım uzaktaki) haber alabilir.

- **Katman 1 (2-hop):** Mesajlaşma bir tur daha tekrarlanınca, C'nin komşuları kendi komşularının bilgisini de almış olduğu için, C artık 2 adım uzaktaki düğümlerin de bilgisine dolaylı olarak erişir.
- **Katman 2 (3-hop):** Ağ daha da genişler. Graf derinleştikçe (katman sayısı arttıkça), her bir düğüm tüm grafın genel topolojisi ve uzak mahalleleri hakkında bilgi sahibi olmaya başlar.

IV. EN ALT BÖLÜM: GNN'lerin Kullanım Alanları

Graf yapılarının gerçek dünyada nerelerde karşımıza çıktığının bir haritasıdır:

- **Sosyal Ağ Analizi:** Topluluk tespiti, viral etki tahmini veya kullanıcı davranış analizi.
- **Öneri Sistemleri:** Kullanıcı-ürün ilişkilerini bir graf olarak modelleyerek (örneğin e-ticaret sitelerinde veya dijital platformlarda) nokta atışı öneriler yapma.
- **Bilgi Grafiği Tamamlama (Knowledge Graph Completion):** Mevcut bir bilgi grafiğindeki eksik ilişkileri veya "bakar körlük" durumundaki gizli bağları tahmin etme.
- **Molekül Özellik Tahmini:** Atomların düğüm, bağların ise kenar olduğu moleküler yapılarda ilaç keşfi ve kimyasal analiz süreçleri.
- **Dolandırıcılık Tespiti:** Finansal işlemleri bir ağ olarak inceleyip şüpheli para transferlerini veya anomali gösteren hesap halkalarını (kara para aklama vb.) yakalama.

Bu görsel, karmaşık graf verilerinin yapay zekâ tarafından nasıl "okunduğunu" ve her bir düğümün çevresiyle konuşarak nasıl anlamlı birer sayısal kimliğe dönüştüğünü gösteren GNN (Graph Neural Networks) mimarisinin bir özetidir.

GNN’lerin temel amacı, bir graf içindeki her düğümün çevresindeki yapıyı dikkate alarak bir temsil (embedding) öğrenmektir. Yani bir düğümün anlamı yalnızca kendi özelliklerinden değil, aynı zamanda bağlı olduğu diğer düğümlerden de türetilir. Bu süreç “mesaj geçirme (message passing)” olarak adlandırılır. Her düğüm, komşularından bilgi alır, bu bilgiyi işler ve kendi temsilini günceller.

Bu yaklaşım, graf yapılarının doğasına son derece uygundur çünkü gerçek dünyadaki birçok sistem ilişkisel yapıdadır. Sosyal ağlar, biyolojik ağlar, ulaşım sistemleri ve bilgi grafları bu tür yapılara örnek olarak verilebilir. GNN’ler bu sistemlerde gizli desenleri ve ilişkileri öğrenebilir.

Bilgi grafı bağlamında GNN’ler özellikle güçlüdür çünkü sadece varlıkları değil, varlıklar arasındaki ilişkileri de öğrenme sürecine dahil eder. Örneğin bir akademik bilgi grafında bir araştırmacının etkisini belirlemek için yalnızca yayın sayısı değil, aynı zamanda işbirliği yaptığı kişiler ve çalışmalar arasındaki bağlantılar da dikkate alınır.

GNN’lerin en önemli özelliklerinden biri yerel bilgiyi küresel bilgiye dönüştürebilme kapasitesidir. Her düğüm yalnızca kendi komşularından bilgi alır, ancak bu süreç katman katman

tekrarlandığında tüm grafın yapısal özellikleri öğrenilebilir hale gelir. Bu, sistemin hem yerel hem de global desenleri yakalamasını sağlar.

Bir diğer önemli avantaj, esneklik ve ölçeklenebilirliktir. GNN'ler farklı büyüklükteki graf yapıları üzerinde çalışabilir ve yeni düğümler eklendiğinde modeli tamamen yeniden eğitmek gerekmez. Bu, dinamik bilgi grafı sistemleri için büyük bir avantajdır.

Ancak GNN'lerin bazı zorlukları da vardır. Özellikle büyük ölçekli graf yapılarında hesaplama maliyeti yüksektir. Ayrıca aşırı düzleşme (over-smoothing) problemi, çok katmanlı GNN'lerde düğüm temsillerinin birbirine çok benzemesine neden olabilir. Bu nedenle model tasarımı dikkatli yapılmalıdır.

Sonuç olarak graf sinir ağları, bilgi grafı tabanlı yapay zekâ sistemlerinin temel öğrenme mekanizmalarından biridir. Bu yapı, ilişkisel veriyi doğrudan öğrenerek yapay zekânın dünyayı bir "graf" olarak algılamasını sağlar. Bu da modern yapay zekânın en önemli yönelimlerinden biri olan "ilişkisel düşünme" paradigmasını güçlendirir.

5.11 Vektör Veri Tabanları ve Graf Veri Tabanları İlişkisi

Yapay zekâ sistemlerinin son yıllardaki gelişimi, veri saklama ve veri temsil etme biçimlerinde iki güçlü yaklaşımı öne çıkarmıştır: vektör veri tabanları ve graf veri tabanları. Bu iki yapı çoğu zaman birbirine alternatif gibi sunulsa da aslında modern yapay zekâ mimarilerinde birbirini tamamlayan iki farklı düşünme biçimini temsil eder. Biri benzerlik üzerinden çalışırken, diğeri ilişki ve yapı üzerinden çalışır.

Vektör veri tabanları, veriyi yüksek boyutlu sayısal uzaylarda temsil eder. Metinler, görseller veya sesler, gömme (embedding) adı verilen sayısal vektörlere dönüştürülür. Bu temsil sayesinde "anlamsal benzerlik" hesaplanabilir. Örneğin iki farklı cümle, kelime düzeyinde farklı olsa bile benzer anlam taşıyorsa vektör uzayında birbirine yakın konumlanır. Bu yaklaşım özellikle büyük dil modelleri (LLM) ile birlikte kullanıldığında güçlü bir semantik arama kapasitesi sağlar.

Python ile bir gömme örneği verelim:

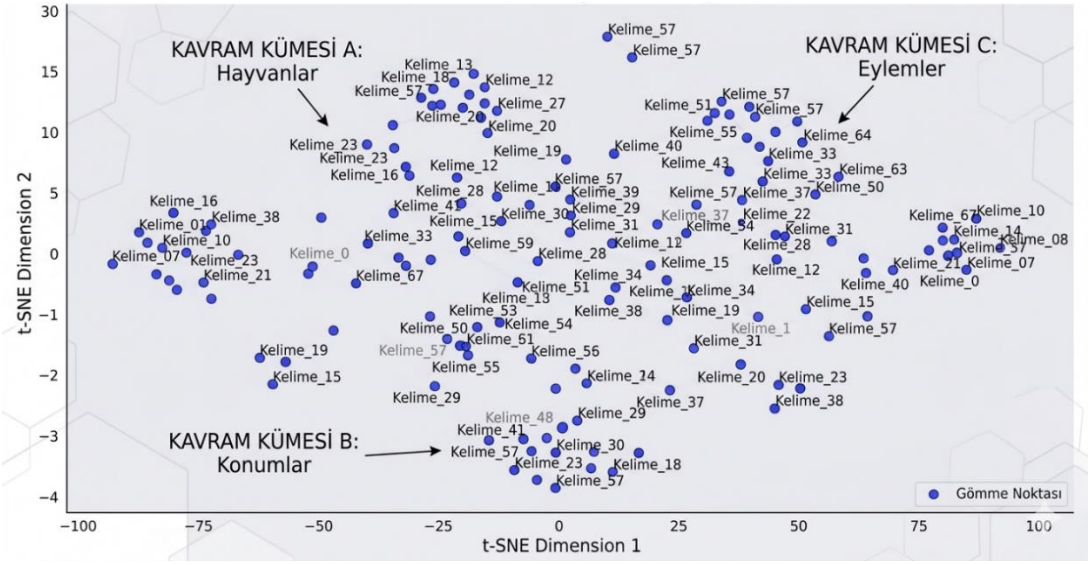
```
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
import numpy as np


# 1. Elinizde 1536 boyutlu gömme vektörleri olduğunu varsayalım
# embeddings = [v1, v2, v3, ...] (Şekli: [Veri_Sayısı, 1536])
embeddings = np.random.rand(100, 1536) # Örnek yapay veri
labels = [f"Kelime_{i}" for i in range(100)]


# 2. t-SNE ile boyutu 1536'dan 2'ye indirgiyoruz
tsne = TSNE(n_components=2, perplexity=15, random_state=42)
embeddings_2d = tsne.fit_transform(embeddings)
```

```
# 3. 2 Boyutlu düzleme çizdiriyoruz
plt.figure(figsize=(10, 8))
plt.scatter(embeddings_2d[:, 0], embeddings_2d[:, 1], c='blue', edgecolors='k')


for i, label in enumerate(labels):
    plt.annotate(label, (embeddings_2d[i, 0], embeddings_2d[i, 1]), alpha=0.7)

plt.title("t-SNE ile Gömme Uzayı Görselleştirmesi")
plt.show()
```

Şekil 19'daki, "t-SNE[11] ile Gömme Uzayı Görselleştirmesi" grafiğini incelediğimizde çok sayıda kelime 57 görüyoruz. Bunun nedenini şöyle açıklayabiliriz: Örneğin, "Kelime_57"nin gerçek hayatta **"YÜZ"** kelimesi olduğunu varsayalım:

- *"Denizde saatlerce **yüz**dü.* Eylemler kümesinde yer alır.
- *"Çocuğun **yüz**ünde bir tebessüm belirdi.*Anatomi/İnsan kümesinde yer alır.
- *"Cüzdanından **yüz** lira çıkardı.*Matematik/Finans kümesinde yer alır.

Şekil 19. t-SNE ile Gömme Uzayı Görselleştirmesi



Eğer bu metinleri gömme modeline sokup t-SNE ile 2 boyutlu düzleme düşürürseniz, aynı "Yüz" (Kelime_57) kelimesi, cümledeki farklı anlamlarından dolayı uzayın farklı yerlerinde, farklı kavram kümelerinin içinde tekrar tekrar karşınıza çıkar.

[11] t-SNE (t-Distributed Stochastic Neighbor Embedding), çok yüksek boyutlu veri kümelerini (örneğin yüzlerce veya binlerce boyuttan oluşan gömme vektörlerini), insan gözünün algılayabileceği 2 veya 3 boyutlu bir düzleme indirmek için kullanılan güçlü bir yapay zeka ve istatistik algoritmasıdır.

Graf veri tabanları ise tamamen farklı bir mantıkla çalışır. Burada temel yapı vektörler değil, varlıklar (düğümler) ve bu varlıklar arasındaki ilişkiler (bağlantılar)dır. Yani bilgi yalnızca "benzerlik" üzerinden değil, "ilişki ağı" üzerinden temsil edilir. Örneğin "Einstein → geliştirdi → Görelilik Teorisi" gibi bir yapı, açık ve yönlü bir bilgi bağlantısıdır. Bu tür ilişkiler, sistemin dünyayı daha yapılandırılmış bir şekilde modellemesini sağlar.

Bu iki yaklaşım arasındaki temel fark, temsil ettikleri "anlam türü"dür. Vektör veri tabanları, anlamı geometrik yakınlık üzerinden ifade ederken; graf veri tabanları, anlamı açık ilişkiler üzerinden ifade eder. Birinde "yakınlık", diğerinde "bağlantı" temel kavramdır. Bu nedenle biri sezgisel benzerlik için, diğeri ise mantıksal ve yapısal analiz için daha uygundur.

Modern yapay zekâ sistemlerinde bu iki yaklaşım giderek daha fazla birlikte kullanılmaktadır. Örneğin bir RAG (Retrieval-Augmented Generation) sisteminde vektör veri tabanı, ilgili belgeleri hızlıca bulmak için kullanılırken, graf veri tabanı bu belgeler arasındaki ilişkileri anlamlandırmak için devreye girer. Böylece sistem hem "neye benzediğini" hem de "nasıl ilişkili olduğunu" anlayabilir.

Graf veri tabanları ayrıca açıklanabilirlik açısından büyük avantaj sağlar. Çünkü her sonuç, belirli bir ilişki yoluyla elde edilir ve bu yol açıkça izlenebilir. Vektör veri tabanlarında ise sonuçlar genellikle "mesafe" üzerinden elde edildiği için açıklama yapmak daha zordur. Bu nedenle kritik karar sistemlerinde graf yapıları daha güvenilir bir temel sunar.

Öte yandan vektör veri tabanları ölçeklenebilirlik ve hız açısından oldukça güçlüdür. Milyonlarca belge arasında hızlı benzerlik araması yapmak mümkündür. Graf veri tabanları ise karmaşık ilişki sorgularında daha güçlüdür ancak çok büyük ölçeklerde performans zorlukları yaşayabilir.

Sonuç olarak vektör veri tabanları ve graf veri tabanları rekabet eden değil, birbirini tamamlayan yapılardır. Vektörler "anlam benzerliğini", graflar ise "anlam ilişkisini" temsil eder. Geleceğin yapay zekâ sistemleri, bu iki yaklaşımı birlikte kullanarak hem sezgisel hem de yapısal düşünme kapasitesine sahip hibrit mimariler üzerine kurulacaktır.

5.12 Yapay Zekânın Belleği Olarak Bilgi Grafı

Yapay zekâ sistemlerinin en temel sınırlamalarından biri kalıcı ve yapılandırılmış bir belleğe sahip olmamalarıdır. Büyük dil modelleri gibi sistemler, büyük miktarda bilgiyle eğitilmiş olsalar da bu bilgi sabit parametreler içinde dağınık bir şekilde bulunur. Bu durum, modelin öğrendiklerini açıkça hatırlamasını, güncellemesini veya organize etmesini zorlaştırır. İşte bu noktada bilgi grafı, yapay zekâ için dışsal bir bellek sistemi olarak devreye girer.

Bilgi grafı, bilgiyi düğümler (varlıklar) ve bağlantılar (ilişkiler) şeklinde yapılandırılmış bir ağ olarak saklar. Bu yapı, insan belleğine oldukça benzer bir organizasyon sunar. İnsanlar bilgiyi tekil veri noktaları olarak değil, ilişkiler ağı içinde hatırlar. Örneğin bir kişiyi hatırlarken yalnızca adını değil, onunla ilgili olayları, yerleri ve ilişkileri de birlikte düşünürüz. Bilgi grafı da benzer şekilde çalışır.

Yapay zekâ açısından bilgi grafının en önemli katkısı, kalıcı ve güncellenebilir bir hafıza sunmasıdır. Modelin eğitimi tamamlandıktan sonra bile bilgi grafı sürekli olarak yeni bilgilerle güncellenebilir. Bu, sistemi statik bir yapıdan çıkarıp dinamik bir bilgi organizmasına dönüştürür. Böylece yapay zekâ, zaman içinde "öğrenmeye devam eden" bir yapıya kavuşur.

Bilgi grafı aynı zamanda bağlam yönetimi açısından da kritik bir rol oynar. Büyük dil modelleri genellikle kısa bağlam pencerelerine sahiptir; yani yalnızca sınırlı miktarda bilgiyi aynı anda işleyebilirler. Bilgi grafı ise uzun vadeli ilişkileri saklayarak bu sınırlamayı aşar. Model, ihtiyaç duyduğunda graf üzerinden ilgili alt yapıyı çekerek bağlam oluşturabilir.

Bir diğer önemli avantaj, bilgi tutarlılığıdır. Dağınık veri kaynaklarında aynı varlık farklı isimlerle veya farklı biçimlerde temsil edilebilir. Bilgi grafı, varlık eşleştirme süreçleri sayesinde bu tutarsızlıkları ortadan kaldırır ve tekil bir bilgi modeli oluşturur. Bu, yapay zekânın daha doğru ve güvenilir sonuçlar üretmesini sağlar.

Ayrıca bilgi grafı, yapay zekâ sistemlerine "hatırlama yeteneği" kazandırır. Bir model geçmişte yaptığı etkileşimleri, kararları ve sonuçları graf üzerinde saklayabilir. Bu sayede sistem yalnızca anlık değil, tarihsel bir perspektife de sahip olur. Bu durum özellikle ajan sistemleri ve uzun vadeli karar destek sistemleri için kritik öneme sahiptir.

Bilgi grafı aynı zamanda açıklanabilirlik açısından da güçlü bir bellek yapısıdır. Bir yapay zekâ sisteminin neden belirli bir sonuca ulaştığı, graf üzerindeki ilişkiler takip edilerek anlaşılabilir. Bu, yapay zekâ ile insan arasında güven ilişkisini güçlendirir.

Sonuç olarak bilgi grafı, yapay zekânın yalnızca bilgi işleyen bir sistem olmaktan çıkıp, bilgiyi hatırlayan, organize eden ve zaman içinde geliştiren bir "bilişsel bellek" yapısına dönüşmesini sağlar. Bu nedenle geleceğin yapay zekâ mimarilerinde bilgi grafı, yalnızca bir veri yapısı değil, sistemin merkezi hafızası olarak konumlanacaktır.

5.13. Üretken Algoritmik Sistemler ve AlphaEvolve

Yapay zekâ araştırmalarının son yıllardaki en önemli dönüşümlerinden biri, sistemlerin yalnızca "öğrenen modeller" olmaktan çıkıp "yeni yöntemler üreten sistemler" haline gelmesidir. Bu dönüşüm, makine öğrenmesinin temel varsayımını değiştirir: artık amaç yalnızca veriden örüntü çıkarmak değil, aynı zamanda daha önce insan tarafından tasarlanan algoritmaları yeniden keşfetmek ve geliştirmektir. Bu yeni yaklaşım, "üretken algoritmik sistemler" kavramı altında toplanabilir.

> Bu sistemler, klasik yapay zekâ uygulamalarından farklı olarak, bir problemi çözmekle yetinmez; çözüm yönteminin kendisini de yeniden üretir. Böylece yapay zekâ, yalnızca bir hesaplama aracı değil, aynı zamanda bir "algoritma keşif motoru" haline gelir.

AlphaEvolve, bu dönüşümün en dikkat çekici örneklerinden biri olarak ortaya çıkmıştır. Onu önemli kılan şey, belirli bir problemi çözmesinden çok, çözüm üretme biçimini sistematik olarak

geliştirebilmesidir. Bu özellik, yapay zekânın bilimsel keşif süreçlerine doğrudan müdahil olabileceği yeni bir dönemin işaretidir.

Kurumsal bağlamda AlphaEvolve, Google DeepMind ekosistemi içinde gelişen araştırma çizgisinin bir devamı olarak değerlendirilebilir. DeepMind'ın AlphaGo ile başlayan stratejik öğrenme, AlphaZero ile genelleşmiş oyun öğrenimi ve AlphaFold ile bilimsel problem çözme alanlarına uzanan evrimi, AlphaEvolve'da algoritma keşfi düzeyine taşınmıştır. Bu açıdan AlphaEvolve, tekil bir ürün olmaktan ziyade, "öğrenmeden keşfe, keşiften algoritma üretimine" uzanan DeepMind araştırma hattının doğal bir uzantısıdır.

Bu bağlamda AlphaEvolve'un temsil ettiği dönüşüm yalnızca teknik değildir; aynı zamanda yapay zekânın bilgi üretim zincirindeki rolünün yeniden tanımlanmasıdır. Sistem artık sadece mevcut algoritmaları optimize eden bir yapı değil, yeni algoritmaların ortaya çıkmasını mümkün kılan bir üretim katmanı olarak konumlanmaktadır.

Algoritma Tasarımından Algoritma Keşfine

Geleneksel bilgisayar bilimlerinde algoritmalar uzun süre boyunca insan zekâsının bilinçli tasarım ürünleri olarak kabul edilmiştir. Bu paradigma içinde algoritma geliştirme süreci, problem tanımı ile başlayan, matematiksel modelleme ile devam eden ve mühendislik sezgisiyle şekillenen doğrusal bir düşünme hattına dayanır. Bir problemi çözmek için önce yapısal bir analiz yapılır, ardından bu yapıya uygun adımlar dizisi tasarlanır ve sonunda bu adımlar programlama dilleri aracılığıyla uygulanabilir hale getirilir. Bu yaklaşımda insan, sistemin hem tasarımcısı hem de nihai karar vericisidir.

Klasik algoritma tasarımında matematiksel sezgi merkezi bir rol oynar. Örneğin bir optimizasyon problemi ele alındığında, çözüm yolu genellikle insan uzmanlığının ürettiği heuristikler üzerinden geliştirilir. Bilgisayar bilimlerinin tarihsel gelişimi de bu çerçeve içinde şekillenmiştir: sıralama algoritmaları, graf algoritmaları, arama teknikleri ve veri yapıları büyük ölçüde insan yaratıcılığının ürünleridir. Her yeni algoritma, belirli bir problem sınıfı için daha verimli bir çözüm üretme çabasının sonucudur.

Makine öğrenmesi ve özellikle derin öğrenme bu yapıyı kısmen değiştirmiştir. Sistemler artık bazı problemler için açıkça tanımlanmış kurallar yerine veriden öğrenilen parametrelerle çalışmaktadır. Ancak bu dönüşüm bile algoritma tasarımının temel doğasını kökten değiştirmemiştir. Çünkü öğrenme sürecinin nasıl gerçekleşeceği, hangi optimizasyon yöntemlerinin kullanılacağı ve model mimarisinin nasıl olacağı hâlâ insan mühendisler tarafından belirlenmektedir. Yani sistem öğrenmekte, fakat öğrenmenin çerçevesi dışarıdan çizilmektedir.

Bu noktada üretken algoritmik sistemler kavramı devreye girer ve klasik yaklaşımı köklü biçimde tersine çevirir. Bu yeni paradigma içinde algoritmalar artık yalnızca insan tarafından yazılan yapılar değildir; aynı zamanda sistematik olarak keşfedilen nesnelerdir. Bu dönüşüm, algoritmayı bir tasarım ürünü olmaktan çıkarıp bir keşif alanına dönüştürür.

Üretken algoritmik sistemlerin temel fikri, olası algoritmaların oluşturduğu geniş bir "çözüm uzayı" tanımlamaktır. Bu uzay içinde her bir nokta, belirli bir problemi çözmeye aday bir algoritmayı temsil eder. Sistem bu uzayı sistematik biçimde tarar, farklı algoritmik varyasyonlar üretir ve bu varyasyonları belirli performans kriterlerine göre değerlendirir. Değerlendirme süreci genellikle hız, doğruluk, kaynak kullanımı veya genelleme kapasitesi gibi ölçütlere dayanır. En iyi performansı gösteren algoritmalar seçilir ve bu süreç iteratif olarak devam eder.

Bu yaklaşım, klasik mühendislik anlayışında radikal bir kırılmaya işaret eder. Çünkü artık insan, tek tek algoritmalar tasarlayan bir özne olmaktan çıkar; bunun yerine algoritmaların evrimleşebileceği bir ekosistem tanımlar. Bu ekosistem içinde sistem, farklı çözüm yollarını deneyerek kendi içinde bir "algoritmik evrim" gerçekleştirir. Bu evrim süreci, doğadaki biyolojik seçilim mekanizmalarına benzer bir yapı sergiler. Ancak burada seçilim baskısı fiziksel çevre değil, hesaplama performansı ve problem çözme başarısıdır.

Bu dönüşümün en önemli sonuçlarından biri, keşif sürecinin ölçeklenebilir hale gelmesidir. İnsan sezgisi belirli bir problem alanında sınırlı sayıda çözüm üretebilirken, üretken sistemler çok daha geniş bir çözüm uzayını kısa sürede tarayabilir. Bu durum, özellikle yüksek boyutlu optimizasyon problemlerinde ve klasik yöntemlerin yetersiz kaldığı alanlarda önemli bir avantaj sağlar.

Bununla birlikte bu yeni paradigma, algoritma kavramının doğasını da yeniden tanımlar. Artık algoritma, sabit ve kesin bir prosedür değil; sürekli evrimleşen, test edilen ve yeniden şekillendirilen dinamik bir yapı haline gelir. Bu bakış açısı, bilgisayar bilimlerini statik tasarımdan dinamik keşfe taşıyan temel dönüşümlerden biridir.

Sonuç olarak algoritma tasarımından algoritma keşfine geçiş, yalnızca teknik bir ilerleme değil, aynı zamanda epistemolojik bir değişimdir. Bilginin üretim biçimi değişmekte, insanın rolü yeniden tanımlanmakta ve yapay zekâ sistemleri giderek daha özerk keşif mekanizmalarına dönüşmektedir. Bu dönüşüm, gelecekte yalnızca daha iyi algoritmalar değil, aynı zamanda algoritma üretmenin kendisine dair yeni bir bilim alanının ortaya çıkacağını göstermektedir.

AlphaEvolve'un Temel Çalışma Prensibi

AlphaEvolve, üretken algoritmik sistemler paradigması içinde evrimsel arama, üretken modelleme ve otomatik değerlendirme mekanizmalarını bir araya getiren hibrit bir keşif sistemi olarak tanımlanabilir. Bu sistemin temel amacı, yalnızca mevcut algoritmaları optimize etmek değil, aynı zamanda algoritmik uzay içinde daha önce keşfedilmemiş çözüm yollarını ortaya çıkarmaktır. Bu yönüyle AlphaEvolve, klasik makine öğrenmesi sistemlerinden ayrılarak, "algoritma tasarımı" yerine "algoritma evrimi" mantığıyla çalışır.

Sistemin çalışma mantığı, iteratif ve döngüsel bir süreç üzerine kuruludur. Bu süreç, sürekli olarak yeni algoritmaların üretilmesi, test edilmesi ve seçilmesi üzerine inşa edilmiştir. Her iterasyon, önceki nesilden öğrenilen bilgiyi kullanarak daha iyi çözümlere ulaşmayı hedefler. Bu yapı, biyolojik evrim süreçlerine benzer bir mantık taşır; ancak burada evrimleşen varlıklar canlı organizmalar değil, matematiksel algoritmalardır.

AlphaEvolve'un temel çalışma döngüsü beş ana aşamadan oluşur:

İlk aşamada, sisteme başlangıç noktası olarak bilinen ve literatürde yer alan mevcut algoritmalar verilir. Bu algoritmalar, belirli bir problem sınıfını çözme kapasitesine sahip, ancak mutlaka optimal olmayan çözümlerden oluşur. Bu aşama, evrimsel sürecin "gen havuzu" olarak düşünülebilir. Yani sistemin başlangıçta sahip olduğu bilgi seti, gelecekte üretilecek tüm varyasyonların temelini oluşturur.

İkinci aşamada sistem, bu mevcut algoritmalar üzerinde küçük ancak sistematik değişiklikler üretir. Bu değişiklikler, algoritmanın yapısal bileşenlerinde, parametrelerinde veya işlem sıralarında olabilir. Üretken modelleme burada devreye girer ve sistem, yalnızca rastgele değişiklikler yapmak yerine, daha önceki performans sonuçlarına dayalı yönlendirilmiş varyasyonlar üretir. Bu durum, klasik rastgele arama yöntemlerinden farklı olarak daha hedefli ve öğrenmeye dayalı bir keşif süreci oluşturur.

Üçüncü aşamada, üretilen her yeni algoritmik varyant farklı problem örnekleri üzerinde test edilir. Bu test süreci, sistemin gerçek performansını ölçmek açısından kritik bir rol oynar. Çünkü bir algoritmanın başarısı yalnızca teorik doğruluğuna değil, farklı veri setleri ve problem senaryoları karşısındaki dayanıklılığına bağlıdır. Bu aşamada sistem, genellikle çok sayıda benchmark üzerinde paralel değerlendirme yaparak geniş bir performans profili oluşturur.

Dördüncü aşamada, performans ölçütlerine göre en iyi çalışan algoritmik varyantlar seçilir. Bu seçim süreci, doğal seçilim mekanizmasına benzer şekilde işler. Daha hızlı, daha doğru veya daha verimli çalışan algoritmalar "hayatta kalırken", düşük performanslı varyantlar elenir. Bu aşama, sistemin evrimsel baskı mekanizmasını oluşturur ve hangi algoritmaların sonraki nesillere aktarılacağını belirler.

Beşinci ve son aşamada ise seçilen algoritmalar yeni nesil üretimin temelini oluşturur. Yani sistem, başarılı varyantları bir sonraki iterasyona başlangıç noktası olarak geri besler. Bu döngü sürekli olarak tekrar eder ve her iterasyonda algoritmik performans kademeli olarak gelişir. Bu süreç, zaman içinde yalnızca küçük iyileştirmeler değil, aynı zamanda yapısal yenilikler de üretebilir.

Bu beş aşamalı döngü, AlphaEvolve'un temel öğrenme ve keşif mekanizmasını oluşturur. Sistem, bu döngü sayesinde sürekli olarak kendi içindeki algoritmik çeşitliliği artırır ve çözüm uzayını genişletir. Böylece yalnızca mevcut çözümleri iyileştirmekle kalmaz, aynı zamanda daha önce var olmayan yeni çözüm stratejileri de ortaya çıkarır.

Bu yapının en dikkat çekici yönlerinden biri, evrimsel sürecin tamamen hesaplamalı bir ortamda gerçekleşmesidir. Doğal evrimde olduğu gibi burada da varyasyon, seçilim ve kalıtım mekanizmaları vardır. Ancak biyolojik sistemlerden farklı olarak, bu süreç tamamen matematiksel performans ölçütlerine dayalıdır. Yani "uygunluk" kavramı, hayatta kalma değil, hesaplama verimliliği ve problem çözme başarısı üzerinden tanımlanır.

Sonuç olarak AlphaEvolve'un çalışma prensibi, algoritmaları statik yapılar olmaktan çıkarıp dinamik, evrimleşen ve sürekli yeniden üretilen sistemler haline getirir. Bu yaklaşım, yapay zekânın yalnızca veri işleyen bir teknoloji olmaktan çıkıp, aynı zamanda bilgi üretim süreçlerini yöneten bir keşif mekanizmasına dönüşmesinin önemli bir örneğini temsil eder.

Matematikteki Başarıları: İnsan Sezgisinin Ötesine Geçmek

AlphaEvolve ve benzeri üretken algoritmik sistemlerin en çarpıcı katkılarından biri, matematiksel problem çözme süreçlerinde insan sezgisinin doğal sınırlarını aşabilme potansiyelidir. Geleneksel matematiksel keşif süreçleri, büyük ölçüde insan sezgisi, deneyimi ve yıllar içinde birikmiş teorik bilgi üzerine kuruludur. Bir araştırmacı yeni bir algoritma önerdiğinde genellikle geçmiş literatüre dayanır, mevcut teknikleri geliştirir veya belirli bir problem sınıfı için sezgisel iyileştirmeler üretir. Ancak bu süreç, kaçınılmaz olarak insan bilişsel kapasitesinin ve hayal gücünün sınırlarıyla çevrilidir.

Üretken algoritmik sistemler ise bu sınırları genişleten yeni bir keşif mekaniği sunar. AlphaEvolve gibi sistemler, çok geniş bir çözüm uzayını sistematik biçimde tarayarak, insan araştırmacıların çoğu zaman fark edemeyeceği ara bölgelerde yeni optimizasyon stratejileri keşfedebilir. Bu durum, özellikle yüksek boyutlu ve karmaşık matematiksel problemlerde belirgin hale gelir. Çünkü bu tür problemler, insan sezgisinin doğrudan görselleştiremeyeceği kadar büyük ve karmaşık yapıdadır.

Özellikle kombinatorik optimizasyon alanı, bu dönüşümün en görünür olduğu alanlardan biridir. Bu alanda problemler genellikle çok sayıda olası çözüm kombinasyonuna sahiptir ve optimal çözümü bulmak hesaplama açısından son derece maliyetlidir. AlphaEvolve benzeri sistemler, bu devasa çözüm uzayını rastgele değil, öğrenmeye dayalı bir şekilde tarayarak daha verimli heuristikler geliştirebilir. Bu sayede bazı durumlarda klasik algoritmaların ulaştığı performans sınırlarının ötesine geçilebilmektedir.

Graf teorisi alanında da benzer bir ilerleme gözlemlenmektedir. En kısa yol problemleri, ağ akışı optimizasyonları ve düğüm merkezilik hesaplamaları gibi klasik problemler, üretken sistemler tarafından farklı bakış açılarıyla yeniden ele alınabilmektedir. Sistem, yalnızca mevcut algoritmaları optimize etmekle kalmaz, aynı zamanda problem temsilini değiştirerek daha düşük hesaplama karmaşıklığına sahip yeni çözüm yolları önerebilir. Bu durum, algoritmik keşfin yalnızca çözüm seviyesinde değil, aynı zamanda problem formülasyonu seviyesinde de gerçekleşebileceğini göstermektedir.

Lineer cebir alanı da bu dönüşümden önemli ölçüde etkilenmektedir. Özellikle matris işlemleri, yüksek performanslı hesaplama sistemlerinin temel bileşenlerinden biridir. AlphaEvolve benzeri sistemlerin bazı deneysel çalışmalarda, matris çarpımı gibi temel işlemler için daha az işlem adımı gerektiren alternatif yöntemler önerebildiği gözlemlenmiştir. Bu tür iyileştirmeler küçük gibi görünse de, büyük ölçekli hesaplamalarda önemli performans kazançları sağlayabilir. Çünkü modern yapay zekâ sistemleri ve bilimsel simülasyonlar, milyonlarca hatta milyarlarca matris işlemi içerebilir.

Bununla birlikte NP-zor problemler, üretken algoritmik sistemlerin en ilginç test alanlarından birini oluşturmaktadır. Bu tür problemler, teorik olarak optimal çözümü bulmanın çok yüksek hesaplama maliyetine sahip olduğu problem sınıflarıdır. Bu nedenle pratikte genellikle yaklaşık çözümler veya sezgisel yöntemler kullanılır. AlphaEvolve benzeri sistemler, bu heuristik yöntemleri geliştirme potansiyeline sahiptir. Özellikle çözüm uzayını farklı açılardan keşfederek, daha önce insan araştırmacıların tasarladığı heuristiklerden daha etkili yaklaşım stratejileri üretebilir.

Bu sonuçların en kritik yönü, sistemin yalnızca mevcut yöntemleri yeniden üretmemesi, aynı zamanda tamamen yeni çözüm bölgelerini keşfedebilmesidir. Matematiksel problem uzayları çoğu zaman yüksek boyutlu ve sezgisel olarak zor yorumlanabilir yapıdadır. İnsan araştırmacılar genellikle bu uzayların belirli bölgelerine yoğunlaşırken, üretken sistemler çok daha geniş bir alanı sistematik olarak tarayabilir. Bu durum, bazı optimal veya yarı-optimal çözümlerin insan sezgisi tarafından hiç fark edilmemiş olabileceği anlamına gelir.

Bu bağlamda AlphaEvolve'un en önemli katkılarından biri, "matematiksel sezgi" kavramına yeni bir boyut kazandırmasıdır. Geleneksel olarak sezgi, insan zihninin deneyim ve soyutlama kapasitesine dayalı bir özellik olarak görülür. Ancak üretken sistemler, büyük veri ve geniş arama uzayları üzerinden insan sezgisine benzer hatta bazı durumlarda onu aşan keşif örüntüleri oluşturabilmektedir. Bu durum, sezginin yalnızca biyolojik bir özellik değil, aynı zamanda hesaplamalı olarak simüle edilebilir bir süreç olabileceği fikrini güçlendirmektedir.

Sonuç olarak AlphaEvolve ve benzeri sistemler, matematiksel keşif süreçlerini yalnızca hızlandırmakla kalmamakta, aynı zamanda bu süreçlerin doğasını da dönüştürmektedir. İnsan sezgisiyle sınırlı olan klasik yaklaşımın yerini, geniş ölçekli hesaplama gücü ve evrimsel arama mekanizmalarıyla desteklenen yeni bir keşif paradigması almaktadır. Bu paradigma, matematiği yalnızca insan aklının ürünü olmaktan çıkarıp, insan ve makine ortak üretimi bir keşif alanına dönüştürmektedir.

Bilimsel ve Mühendislik Alanlarındaki Etkiler

AlphaEvolve benzeri üretken sistemler yalnızca soyut matematikte değil, mühendislik ve bilimsel modelleme alanlarında da etkiler göstermeye başlamıştır.

Özellikle şu alanlarda potansiyel etkiler gözlemlenmektedir:

- Optimizasyon problemleri: Enerji dağıtımı, lojistik planlama ve kaynak yönetimi gibi alanlarda daha verimli algoritmalar
- Makine öğrenmesi altyapısı: Model eğitim süreçlerinde daha hızlı veya daha stabil optimizasyon yöntemleri
- Simülasyon sistemleri: Fiziksel süreçlerin daha düşük maliyetle simüle edilmesi
- Veri işleme boru hatları: Büyük ölçekli veri sistemlerinde daha verimli akış algoritmaları

Bu alanlarda sistemlerin ortak özelliği, insan tarafından uzun yıllar içinde geliştirilen heuristiklerin ötesine geçebilme potansiyelidir.

Üretken Sistemlerin Çalışma Mantığı: Arama Uzayı ve Seçilim

AlphaEvolve'un temel gücü, çok geniş bir "algoritma uzayı" içinde arama yapabilmesidir. Bu uzayda her algoritma bir aday çözüm olarak temsil edilir. Sistem bu adaylar üzerinde sürekli varyasyon üretir.

Bu süreç üç temel bileşene dayanır:

(1) Varyasyon üretimi
Mevcut algoritmalar küçük değişikliklerle yeni adaylara dönüştürülür.

(2) Değerlendirme
Her aday algoritma belirli benchmark problemleri üzerinde test edilir.

(3) Seçilim
En iyi performans gösteren algoritmalar korunur ve yeni nesil üretimin temeli olur.

Bu yapı, evrimsel biyolojideki doğal seçilim mekanizmasına matematiksel bir karşılık oluşturur.

AlphaEvolve ve İnsan Bilgisinin Sınırları

AlphaEvolve ve benzeri üretken algoritmik sistemlerin ortaya çıkışı, yalnızca teknik bir ilerlemeyi değil, aynı zamanda insan bilgisinin doğasına ilişkin temel varsayımların yeniden sorgulanmasını da beraberinde getirmiştir. Geleneksel bilim ve mühendislik anlayışı, uzun süre boyunca algoritmik tasarımın insan zekâsına özgü bir faaliyet olduğunu varsaymıştır. Bu perspektife göre insan, problem alanını tanımlar, çözüm uzayını zihinsel olarak sınırlar ve ardından en uygun algoritmayı sezgisel, analitik ve deneysel yöntemlerle geliştirir. Ancak AlphaEvolve gibi sistemler bu varsayımı doğrudan zorlamaktadır.

Bu tür sistemlerin en önemli felsefi sonucu, insan bilgisinin algoritmik tasarımda mutlak bir sınır olmadığı fikrini görünür hale getirmesidir. Çünkü üretken algoritmik sistemler, insan araştırmacıların çoğu zaman fark edemeyeceği çözüm bölgelerini keşfedebilmekte ve bu bölgelerde daha önce düşünülmemiş algoritmik yapıların ortaya çıkmasına aracılık edebilmektedir. Bu durum, problem çözme sürecinin yalnızca insan sezgisine bağlı olmadığını, aynı zamanda geniş ölçekli hesaplama ve sistematik arama mekanizmalarıyla da yürütülebileceğini göstermektedir.

Bu noktada temel kırılma, "algoritma üretimi" kavramının doğasında ortaya çıkar. İnsan merkezli yaklaşımda algoritma üretimi yaratıcı bir zihinsel faaliyet olarak kabul edilirken, üretken sistemlerde bu süreç büyük ölçüde otomatikleştirilebilir bir keşif problemine dönüşmektedir. Sistem, farklı algoritmik varyasyonları üretir, bunları test eder ve performans kriterlerine göre seçer. Bu süreç tekrarlandıkça, insanın doğrudan tasarlamadığı ancak sistematik olarak ortaya çıkan yeni algoritmalar oluşur. Bu durum, algoritmik yaratıcılığın yalnızca insan zekâsına özgü bir yetenek olmadığı fikrini güçlendirir.

Bu gelişme iki önemli sonucu beraberinde getirir. İlk olarak, algoritmik keşif artık insan zekâsının tekeli olmaktan çıkmaktadır. Bu, bilgi üretim süreçlerinde yeni bir güç dağılımına işaret eder. Eskiden yalnızca insan araştırmacıların geliştirebildiği algoritmalar, artık büyük ölçekli hesaplama sistemleri tarafından da üretilebilmektedir. Bu değişim, bilimsel üretimin merkezinde yer alan "insan yaratıcılığı" kavramının yeniden tanımlanmasını gerektirir. Yaratıcılık, artık yalnızca bireysel sezgiye dayalı bir süreç değil, aynı zamanda sistematik keşif mekanizmalarıyla desteklenen kolektif ve hesaplamalı bir süreç haline gelmektedir.

İkinci önemli sonuç ise matematiksel ve mühendislik bilgi üretiminin giderek otomatikleşmeye başlamasıdır. AlphaEvolve benzeri sistemler, yalnızca mevcut bilgiyi optimize etmekle kalmaz; aynı zamanda yeni bilgi yapılarını ortaya çıkarabilir. Bu durum, bilimsel keşif süreçlerinin hızını ve ölçeğini kökten değiştirme potansiyeline sahiptir. Özellikle karmaşık sistemlerde, insan araştırmacıların ulaşmakta zorlandığı çözüm uzayları, üretken algoritmik sistemler tarafından daha sistematik ve geniş kapsamlı bir şekilde taranabilmektedir.

Bu dönüşüm, bilimsel üretim süreçlerinin geleceği açısından kritik bir kırılma noktasıdır. Çünkü bilgi üretimi artık yalnızca insan zihninin sınırlı kapasitesiyle değil, aynı zamanda algoritmik sistemlerin geniş arama ve optimizasyon yetenekleriyle şekillenmektedir. Bu durum, bilimsel keşif sürecinin doğasını değiştirerek onu daha hibrit bir yapıya dönüştürmektedir: İnsan sezgisi ile makine hesaplamasının birlikte çalıştığı yeni bir epistemolojik model ortaya çıkmaktadır.

Sonuç olarak AlphaEvolve, yalnızca bir algoritma üretim sistemi değil, aynı zamanda insan bilgisinin sınırlarına ilişkin klasik varsayımları yeniden düşünmeye zorlayan bir dönüm noktasıdır. Bu sistemler, bilginin üretiminde insanın rolünü ortadan kaldırmaz; ancak bu rolü yeniden tanımlar. İnsan artık tek başına keşfeden değil, keşif süreçlerini yönlendiren, çerçeveleyen ve denetleyen bir aktör haline gelir. Bu değişim, yapay zekâ çağında bilginin nasıl üretileceğine dair en temel sorulardan birini gündeme getirir: Eğer algoritmalar kendi kendilerini geliştirebiliyorsa, insan zekâsının rolü nerede başlar ve nerede biter?

Bilgi Grafı Perspektifi: Yapıdan Üretime Geçiş

Bu kitabın temel ekseni olan bilgi grafı yaklaşımı açısından bakıldığında, AlphaEvolve ve benzeri üretken algoritmik sistemler yalnızca teknik bir yenilik değil, aynı zamanda bilgi temsili paradigmasında köklü bir dönüşümün işaretidir. Bilgi grafı sistemleri uzun süredir dünyayı düğümler (varlıklar) ve kenarlar (ilişkiler) üzerinden modelleyen yapılar olarak kullanılmaktadır. Bu yaklaşımın temel amacı, gerçek dünyadaki karmaşık ilişkileri daha okunabilir, sorgulanabilir ve hesaplanabilir bir yapıya dönüştürmektir. Örneğin bir kişi, çalıştığı kurum, ürettiği eserler ve içinde bulunduğu sosyal ağlar arasındaki bağlantılar bir graf yapısı içinde temsil edilebilir. Böylece bilgi, yalnızca veri noktalarından değil, bu veri noktaları arasındaki ilişkilerden anlam kazanır.

Ancak üretken algoritmik sistemlerin ortaya çıkışı, bu temsil odaklı yaklaşımı bir üst seviyeye taşımaktadır. AlphaEvolve gibi sistemler artık yalnızca mevcut bilgi yapılarını modellemekle kalmaz; aynı zamanda bu yapıların üzerinde yeni algoritmik düzenler ve çözüm mekanizmaları

üretir. Bu durum, bilgi grafı yaklaşımının pasif bir temsil sisteminden aktif bir üretim sistemine doğru evrilmesine işaret eder.

> Bu noktada temel bir ayrım ortaya çıkar. Geleneksel bilgi grafı sistemleri dünyayı temsil ederken, üretken algoritmik sistemler dünyayı yeniden kurabilecek algoritmalar üretme kapasitesine sahiptir. Bilgi grafı, gerçek dünyadaki nesneler ve ilişkiler arasındaki mevcut yapıyı yakalamaya çalışır. Yani onun işlevi tanımlamak, organize etmek ve görünür kılmaktır. Buna karşılık AlphaEvolve benzeri sistemler, bu yapı üzerinde yeni ilişkiler, yeni kurallar ve yeni çözüm yolları üretebilir. Başka bir ifadeyle, bilgi grafı "olanı" modellemeye odaklanırken, üretken algoritmik sistemler "olabilecek olanı" keşfetmeye yönelir.

Bu ayrım, bilgi teknolojilerinde temel bir paradigma kaymasına işaret eder. Çünkü bilgi artık yalnızca temsil edilen bir nesne değil, aynı zamanda aktif olarak üretilen bir süreç haline gelmektedir. Bilgi grafı sistemleri bir anlamda dünyanın haritasını çıkarırken, üretken algoritmik sistemler bu harita üzerinde yeni yollar, yeni bağlantılar ve hatta yeni yapılar inşa etmeye başlamaktadır. Bu durum, bilginin statik bir yapı olmaktan çıkıp dinamik bir evrim sürecine dönüşmesini sağlar.

AlphaEvolve'un temsil ettiği yaklaşım, bu bağlamda bilgi grafı düşüncesine yeni bir boyut kazandırır. Artık mesele yalnızca varlıklar arasındaki ilişkileri modellemek değildir; aynı zamanda bu ilişkilerden yeni algoritmik yapılar türetmektir. Bu, bilgi grafının yalnızca bir veri yapısı değil, aynı zamanda bir üretim zemini haline gelmesi anlamına gelir. Başka bir deyişle, graf yapısı artık sadece dünyayı açıklayan bir araç değil, dünyayı yeniden şekillendiren bir hesaplama alanına dönüşmektedir.

Bu dönüşüm aynı zamanda bilgi üretiminin doğasını da değiştirir. Geleneksel yaklaşımda bilgi insan zihninde üretilir ve daha sonra bir yapıya dökülür. Ancak üretken algoritmik sistemlerde bilgi, doğrudan sistemin içinde evrimleşir. Sistem, mevcut bilgi grafı üzerinde yeni bağlantılar dener, bu bağlantıları değerlendirir ve en iyi performans gösteren yapıların üzerine inşa eder. Bu süreç, bilginin kendisini sürekli yeniden üreten bir döngü oluşturur.

Sonuç olarak AlphaEvolve ve benzeri sistemler, bilgi grafı yaklaşımını statik bir temsil modelinden dinamik bir üretim modeline dönüştürmektedir. Bu dönüşüm, bilgi temsili ile bilgi üretimi arasındaki sınırın giderek bulanıklaştığını göstermektedir. Artık bilgi yalnızca "ne olduğunu" açıklayan bir yapı değil, aynı zamanda "ne olabileceğini" üreten bir süreç haline gelmiştir. Bu da yapay zekâ çağında bilginin doğasına ilişkin en temel değişimlerden birini temsil eder.

RSI ile İlişki: Tam Değil, Ama Yakın

AlphaEvolve doğrudan bir Kendini Yinelemeli Geliştirme (RSI) sistemi değildir. Çünkü:

- kendi öğrenme algoritmasını değiştirmez
- hedef fonksiyon dışarıdan tanımlanır
- sistemin mimarisi sabittir

Ancak buna rağmen RSI'ye önemli bir geçiş katmanı oluşturur. Çünkü sistem:

- yeni algoritmalar üretir
- çözüm uzayını genişletir
- keşif süreçlerini otomatikleştirir

Bu nedenle AlphaEvolve, RSI'nin kendisi değil, RSI'ye giden "üretken ara katman" olarak değerlendirilebilir.

Sonuç: Algoritmaların Evrim Çağı

AlphaEvolve ve benzeri üretken algoritmik sistemler, yapay zekânın evriminde yeni bir dönemi temsil eder. Bu dönem, yalnızca veri işleyen sistemlerden, yeni yöntemler üreten sistemlere geçişi ifade eder. Bu dönüşümün uzun vadeli sonucu şudur: Algoritmalar artık sabit insan icatları değildir; evrimleşen, keşfedilen ve sürekli yeniden üretilen yapılardır.

Bu bağlamda yapay zekâ, yalnızca problemleri çözen bir araç olmaktan çıkar ve problem çözme biçimlerini de üreten bir sisteme dönüşür. Ve bu dönüşüm, bizi daha büyük bir soruya götürür: Eğer algoritmalar kendi kendilerini geliştirebiliyorsa, bilimsel keşfin sınırı nerede duracaktır?

BÖLÜM 6 — BİLGİ GRAFI OLUŞTURMA TEKNİKLERİ

Bu bölümde, bilgi grafı sistemlerinin yalnızca teorik bir kavram olmadığını, aynı zamanda pratik olarak nasıl inşa edildiğini ele alacağız. Bilgi grafı oluşturma süreci, ham verinin anlamlı, ilişkisel ve sorgulanabilir bir yapıya dönüştürülmesini kapsar. Bu süreç, modern yapay zekâ ve veri bilimi uygulamalarının en kritik aşamalarından biridir çünkü verinin nasıl temsil edildiği, sistemin neyi "anladığını" doğrudan belirler.

Gerçek dünyada veri çoğunlukla yapılandırılmamış biçimde bulunur. Metin belgeleri, PDF dosyaları, web sayfaları, sosyal medya içerikleri ve akademik makaleler bu kategoriye girer. Bu veriler doğrudan analiz edilemez; önce içlerindeki varlıkların (kişiler, kurumlar, kavramlar) ve bu varlıklar arasındaki ilişkilerin çıkarılması gerekir. İşte bilgi grafı oluşturma teknikleri bu dönüşüm sürecini yönetir.

Bu bölümde ilk olarak yapılandırılmamış veriden bilgi çıkarma süreçleri incelenecektir. Ardından doğal dil işleme (NLP) teknikleri kullanılarak varlık çıkarımı ve ilişki çıkarımı yöntemleri ele alınacaktır. Bu iki adım, bilgi grafının temel yapı taşlarını oluşturur. Daha sonra gömme tabanlı yöntemlerle daha derin anlamsal ilişkilerin nasıl yakalanabileceği açıklanacaktır.

Ayrıca veri temizleme, bilgi füzyonu ve otomatik ontoloji üretimi gibi kritik aşamalar da bu bölümün kapsamındadır. Gerçek dünya verisi genellikle gürültülüdür ve tutarsızlık içerir; bu nedenle güvenilir bir bilgi grafı oluşturmak için bu verinin normalize edilmesi gerekir.

Son olarak, büyük dil modellerinin (LLM) bilgi grafı inşasında nasıl kullanılabileceği ve PDF, web veya sosyal medya kaynaklarından otomatik bilgi çıkarımının nasıl yapılabileceği ele alınacaktır. Bu bölüm, okuyucuya ham veriden çalışan bir bilgi grafına giden tüm süreci adım adım gösteren pratik bir çerçeve sunmayı amaçlamaktadır.

6.1 Yapılandırılmamış Veriden Bilgi Çıkarma

Yapılandırılmamış veriden bilgi çıkarma, bilgi grafı oluşturma sürecinin en temel ve en zor aşamalarından biridir. Çünkü gerçek dünya verisinin büyük bir kısmı düzenli tablolar veya standart veri şemaları halinde değil, serbest metin biçiminde bulunur. PDF dosyaları, akademik makaleler, e-postalar, haber metinleri, sosyal medya içerikleri ve web sayfaları bu kategoriye girer. Bu tür veriler insan tarafından okunabilir olsa da makine tarafından doğrudan anlaşılabilir değildir. Dolayısıyla ilk adım, bu ham verinin anlamlı bir yapıya dönüştürülmesidir.

Bu süreçte temel amaç, metin içindeki “varlıkları” ve bu varlıklar arasındaki “ilişkileri” tespit etmektir. Varlıklar; kişiler, kurumlar, yerler, olaylar, tarihler veya kavramlar olabilir. İlişkiler ise bu varlıklar arasındaki bağlantıları ifade eder. Örneğin bir metinde “Albert Einstein, 1905 yılında özel görelilik teorisini yayımladı” cümlesi geçiyorsa, burada “Albert Einstein” bir varlık, “özel görelilik teorisi” bir başka varlık ve “yayımladı” bu iki varlık arasındaki ilişkidir.

Yapılandırılmamış veriden bilgi çıkarma süreci genellikle birkaç aşamada gerçekleşir. İlk aşama veri temizleme ve ön işleme aşamasıdır. Bu aşamada gereksiz karakterler, format hataları ve dilsel gürültüler ortadan kaldırılır. İkinci aşama metnin parçalara ayrılmasıdır (tokenization). Bu işlem, metni daha küçük ve işlenebilir birimlere böler.

Üçüncü aşama ise anlamsal analizdir. Bu aşamada metnin sadece kelime düzeyinde değil, anlam düzeyinde de analiz edilmesi gerekir. Burada doğal dil işleme (NLP) teknikleri devreye girer. Dil modelleme, adlandırılmış varlık tanıma ve bağıntı çıkarımı gibi yöntemler kullanılarak metnin içindeki bilgi yapısı ortaya çıkarılır.

Bu sürecin en kritik zorluklarından biri belirsizliktir. Aynı kelime farklı bağlamlarda farklı anlamlara gelebilir. Örneğin “Apple” kelimesi bir meyveyi veya bir teknoloji şirketini ifade edebilir. Bu tür durumlarda bağlam analizi yapılmadan doğru bilgi çıkarımı mümkün değildir.

Bir diğer önemli sorun ise bilgi parçalanmasıdır. Aynı varlık farklı metinlerde farklı biçimlerde ifade edilebilir. Örneğin “IBM”, “International Business Machines” veya “IBM Corp.” aynı kurumu ifade edebilir. Bu nedenle varlık eşleştirme süreçleri büyük önem taşır.

Modern sistemlerde bu süreç giderek daha otomatik hale gelmiştir. Büyük dil modelleri (LLM) ve gömme tabanlı yöntemler, metin içindeki anlam ilişkilerini daha doğru bir şekilde yakalayabilmektedir. Ancak yine de tamamen otomatik ve hatasız bir sistem oluşturmak zordur. Bu nedenle insan denetimi ve geri bildirim mekanizmaları hala kritik bir rol oynar.

Sonuç olarak yapılandırılmamış veriden bilgi çıkarma, ham ve düzensiz veriyi anlamlı bir bilgi grafına dönüştürmenin ilk ve en önemli adımıdır. Bu süreç, yalnızca teknik bir veri işleme aşaması değil, aynı zamanda veriye anlam kazandırma sürecidir.

6.2 Doğal Dil İşleme ile Varlık Çıkarımı

Doğal Dil İşleme (Natural Language Processing, NLP) ile varlık çıkarımı, metin içindeki önemli nesnelerin, kişilerin, kurumların, yerlerin ve kavramların otomatik olarak tespit edilmesi sürecidir. Bu süreç, bilgi grafı oluşturmanın temel yapı taşlarından biridir çünkü bir grafın düğümleri genellikle bu varlıklardan oluşur. Dolayısıyla doğru ve tutarlı bir varlık çıkarımı, tüm bilgi grafının kalitesini doğrudan belirler.

Varlık çıkarımı, genellikle "Adlandırılmış Varlık Tanıma" (Named Entity Recognition, NER) yöntemiyle gerçekleştirilir. Bu yöntem, metin içindeki özel isimleri tanımlayarak bunları belirli kategorilere ayırır. Örneğin bir cümlede geçen "Paris", "Fransa" ve "Seine Nehri" gibi ifadeler sırasıyla şehir, ülke ve coğrafi varlık olarak sınıflandırılabilir.

Bu süreçte kullanılan yöntemler tarihsel olarak büyük bir evrim geçirmiştir. İlk dönemlerde kural tabanlı sistemler kullanılırken, daha sonra istatistiksel modeller ve makine öğrenmesi yöntemleri geliştirilmiştir. Günümüzde ise derin öğrenme ve özellikle transformer tabanlı büyük dil modelleri, varlık çıkarımında en yüksek başarıyı sağlayan yöntemler haline gelmiştir.

Varlık çıkarımının en önemli zorluklarından biri bağlam bağımlılığıdır. Aynı kelime farklı bağlamlarda farklı varlık türlerini ifade edebilir. Örneğin "Washington" kelimesi bir kişi, bir şehir veya bir eyalet olabilir. Bu tür durumlarda modelin doğru çıkarım yapabilmesi için çevresel bağlamı analiz etmesi gerekir.

Bir diğer önemli konu çoklu dil ve çoklu ifade problemidir. Aynı varlık farklı dillerde farklı isimlerle ifade edilebilir. Örneğin "Germany", "Almanya" ve "Deutschland" aynı ülkeyi ifade eder. Bu nedenle çok dilli sistemlerde varlık normalizasyonu (entity normalization) büyük önem taşır.

Modern NLP tabanlı varlık çıkarımı sistemleri genellikle gömme temelli temsil yöntemlerini kullanır. Bu yöntemlerde kelimeler ve ifadeler sayısal vektörlere dönüştürülür ve bu vektörler üzerinden anlamsal benzerlikler hesaplanır. Böylece yalnızca yüzeysel eşleşme değil, derin anlam ilişkileri de yakalanabilir.

Varlık çıkarımı tek başına yeterli değildir; genellikle ilişki çıkarımı ile birlikte kullanılır. Çünkü bir bilgi grafı yalnızca varlıklardan değil, bu varlıklar arasındaki bağlantılardan oluşur. Ancak doğru varlık tespiti olmadan doğru ilişki kurulması da mümkün değildir. Bu nedenle varlık çıkarımı, tüm bilgi grafı inşa sürecinin temel başlangıç noktasıdır.

Sonuç olarak NLP ile varlık çıkarımı, doğal dildeki ham metni yapılandırılmış bilgiye dönüştüren kritik bir adımdır. Bu süreç sayesinde metinler yalnızca okunabilir içerikler olmaktan çıkar, makine tarafından işlenebilir bilgi yapılarına dönüşür. Bu da modern yapay zekâ sistemlerinin temel bilgi üretim mekanizmasını oluşturur.

6.3 İlişki Çıkarımı

İlişki çıkarımı (relation extraction), doğal dilde yer alan metinlerden varlıklar arasındaki anlamlı bağlantıların otomatik olarak tespit edilmesi sürecidir. Bilgi grafı inşa sürecinde varlıklar düğümleri temsil ederken, ilişkiler bu düğümleri birbirine bağlayan bağlantıları oluşturur. Bu nedenle ilişki çıkarımı, yalnızca bir metni anlamlandırma aşaması değil, aynı zamanda bir bilgi mimarisi kurma aşamasıdır.

Örneğin "Albert Einstein, 1905 yılında özel görelilik teorisini geliştirdi" cümlesinde iki varlık vardır: "Albert Einstein" ve "özel görelilik teorisi". Bu iki varlık arasındaki ilişki ise "geliştirdi" fiilidir. Ancak ilişki çıkarımı yalnızca fiilleri bulmak değildir; asıl amaç, bu ilişkilerin türünü, yönünü ve anlamını doğru şekilde modellemektir. Yani bu örnekte ilişki "geliştirmek" değil, daha üst düzeyde "bilimsel katkı sağlamak" gibi bir semantik kategoriye de bağlanabilir.

İlişki çıkarımı süreci genellikle varlık çıkarımından sonra gelir. Çünkü ilişkiler, varlıklar olmadan tanımlanamaz. Bu nedenle NLP tabanlı sistemlerde önce adlandırılmış varlıklar tespit edilir, ardından bu varlıklar arasındaki olası ilişkiler analiz edilir. Bu analiz sırasında cümle yapısı, sözdizimsel bağımlılıklar ve anlamsal bağlam birlikte değerlendirilir.

Geleneksel yöntemlerde ilişki çıkarımı için kural tabanlı sistemler kullanılmıştır. Bu sistemler belirli dil kalıplarını tanımlayarak ilişkileri yakalamaya çalışır. Ancak bu yaklaşım esnek değildir ve farklı ifade biçimlerine karşı zayıf kalır. Örneğin aynı ilişki "geliştirdi", "ortaya koydu" veya "formüle etti" gibi farklı kelimelerle ifade edilebilir. Kural tabanlı sistemler bu çeşitliliği yakalamakta zorlanır.

Makine öğrenmesi ve özellikle derin öğrenme yöntemleri, ilişki çıkarımında büyük bir dönüşüm yaratmıştır. Bu modeller, büyük miktarda etiketli veri üzerinden öğrenerek varlıklar arasındaki ilişkileri otomatik olarak sınıflandırabilir. Transformer tabanlı modeller, bağlamı daha iyi analiz ederek daha karmaşık ilişki yapılarını yakalayabilir hale gelmiştir.

İlişki çıkarımında en büyük zorluklardan biri belirsizliktir. Bir cümlede birden fazla ilişki olasılığı bulunabilir ve modelin doğru olanı seçmesi gerekir. Ayrıca bazı ilişkiler açıkça ifade edilmez, örtük olarak metin içinde yer alır. Örneğin "Einstein'ın teorisi modern fiziği değiştirdi" cümlesinde "Einstein" ile "modern fizik" arasında dolaylı bir etki ilişkisi vardır.

Bir diğer önemli konu ilişki türlerinin sınıflandırılmasıdır. Tüm ilişkiler aynı değildir; "çalışır", "sahiptir", "doğmuştur", "öğrencisidir" gibi farklı ilişki türleri vardır. Bu türlerin doğru şekilde etiketlenmesi, bilgi grafının anlamsal zenginliğini belirler.

Modern sistemlerde ilişki çıkarımı genellikle gömme tabanlı yöntemlerle desteklenir. Bu yöntemler sayesinde yalnızca yüzeysel dil kalıpları değil, kelimelerin anlamsal yakınlıkları da dikkate alınır. Böylece daha esnek ve genellenebilir modeller oluşturulur.

Sonuç olarak ilişki çıkarımı, bilgi grafı oluşturma sürecinin en kritik aşamalarından biridir. Çünkü bilgi grafının gerçek değeri, varlıkların kendisinden çok bu varlıklar arasındaki ilişkilerde ortaya

çıkar. İyi çıkarılmış ilişkiler, bir metni yalnızca okunabilir bir içerik olmaktan çıkarır ve onu yapısal bir bilgi sistemine dönüştürür.

6.4 Gömme (Embedding) Tabanlı Bilgi Çıkarımı

Gömme (embedding) tabanlı bilgi çıkarımı, metinlerin, varlıkların ve ilişkilerin sayısal vektör uzayında temsil edilerek anlamsal yapıların otomatik olarak öğrenilmesini sağlayan modern bir yöntemdir. Bu yaklaşım, klasik kural tabanlı veya tamamen sözdizimsel yöntemlerin ötesine geçerek, anlamı matematiksel bir uzayda modellemeye dayanır.

Gömme kavramının temel fikri, benzer anlamlara sahip öğelerin vektör uzayında birbirine yakın konumlanmasıdır. Örneğin “doktor” ve “hekim” kelimeleri farklı yazıma sahip olsalar da benzer anlam taşıdıkları için vektör uzayında birbirine yakın yer alırlar. Bu özellik, bilgi çıkarımı süreçlerinde büyük bir avantaj sağlar çünkü sistem yalnızca kelime eşleşmesine değil, anlam benzerliğine de dayanır.

Gömme tabanlı bilgi çıkarımı özellikle büyük dil modelleri (LLM) ve transformer mimarileri ile birlikte güçlü bir hale gelmiştir. Bu modeller, kelimeleri ve cümleleri yüksek boyutlu vektörlere dönüştürerek metnin anlamsal haritasını çıkarır. Bu harita üzerinde yapılan işlemler sayesinde varlıklar, ilişkiler ve hatta gizli bağlamlar tespit edilebilir.

Bu yöntemin en önemli kullanım alanlarından biri semantik aramadır. Geleneksel arama sistemleri anahtar kelime eşleşmesine dayanırken, gömme tabanlı sistemler anlam benzerliğini dikkate alır. Böylece kullanıcı “kalp hastalıkları tedavisi” gibi bir sorgu yaptığında, tam olarak aynı kelimeleri içermeyen ancak benzer anlam taşıyan belgeler de sonuçlara dahil edilir.

Bilgi grafı bağlamında gömme yöntemleri, yalnızca metin düzeyinde değil, graf yapıları üzerinde de kullanılabilir. Düğüm gömme adı verilen tekniklerde, graf içindeki düğümler vektör uzayına taşınır. Böylece bir düğümün anlamı yalnızca kendi içeriğiyle değil, graf içindeki konumuyla birlikte temsil edilir.

İlişki çıkarımı açısından gömme yöntemleri, örtük ilişkileri ortaya çıkarmada oldukça etkilidir. Çünkü bazı ilişkiler metinde açıkça belirtilmez. Örneğin iki bilim insanı aynı araştırma alanında çalışıyor olabilir ancak doğrudan birlikte çalıştıkları belirtilmemiş olabilir. Gömme uzayında bu iki varlık birbirine yakın konumlanarak potansiyel bir ilişkiyi işaret eder.

Bu yaklaşımın bir diğer önemli avantajı genelleme kapasitesidir. Model, daha önce görmediği kelime veya kavramları bile benzerlik ilişkileri üzerinden anlamlandırabilir. Bu, özellikle dinamik ve sürekli büyüyen veri setlerinde büyük bir avantaj sağlar.

Ancak gömme tabanlı yöntemlerin de bazı sınırlamaları vardır. En önemli sınırlama açıklanabilirlik problemidir. Vektör uzayında yakınlık ilişkileri güçlü bir sinyal sunar, ancak bu ilişkinin neden oluştuğunu açıklamak zordur. Bu nedenle gömme yöntemleri genellikle bilgi grafı gibi açık yapılarla birlikte kullanılır.

Sonuç olarak gömme tabanlı bilgi çıkarımı, modern yapay zekâ sistemlerinde anlamın sayısallaştırılması için en güçlü yöntemlerden biridir. Bu yaklaşım, metni yalnızca bir kelime dizisi olarak değil, çok boyutlu bir anlam uzayı olarak ele alır. Bilgi grafi ile birlikte kullanıldığında ise hem yapısal hem de anlamsal açıdan güçlü hibrit sistemler oluşturulmasını sağlar.

6.5 Bilgi Füzyonu

Bilgi füzyonu, farklı kaynaklardan gelen verilerin tek ve tutarlı bir bilgi yapısı içinde birleştirilmesi sürecidir. Bilgi grafi oluşturma bağlamında bu süreç, aynı varlık veya ilişki hakkında farklı sistemlerde bulunan bilgilerin uyumlu hale getirilmesini ifade eder. Gerçek dünyada veri çoğu zaman parçalıdır; aynı kişi, kurum veya olay farklı veri tabanlarında farklı biçimlerde temsil edilir. Bu durum, bilgi üretiminde hem tutarsızlık hem de belirsizlik yaratır. Bilgi füzyonu, bu parçalı yapıyı bütüncül bir bilgi modeline dönüştürmeyi amaçlar.

Bilgi füzyonunun temel problemi “aynı şeyin farklı şekilde ifade edilmesi”dir. Örneğin bir veri kaynağında “IBM”, başka bir kaynakta “International Business Machines”, bir diğerinde ise “IBM Corp.” şeklinde geçen ifadeler aslında aynı varlığı temsil eder. Ancak bu eşleştirme yapılmadığında bilgi grafi içinde gereksiz tekrarlar ve çelişkili yapılar oluşur. Bu nedenle ilk adım varlık eşleştirme süreçlerinin doğru şekilde uygulanmasıdır.

Bilgi füzyonu yalnızca varlıkları birleştirmekle sınırlı değildir; aynı zamanda ilişkilerin de birleştirilmesini kapsar. Farklı kaynaklar aynı ilişkiyi farklı biçimlerde tanımlayabilir. Örneğin “çalışır”, “görev yapar”, “istihdam edilir” gibi ifadeler aynı anlam kümesine ait olabilir. Bu ilişkilerin doğru şekilde normalize edilmesi gerekir.

Modern bilgi sistemlerinde bilgi füzyonu çok katmanlı bir süreçtir. İlk katmanda veri kaynakları toplanır ve ön işleme tabi tutulur. İkinci katmanda varlık ve ilişki çıkarımı yapılır. Üçüncü katmanda ise bu çıkarımlar arasında eşleştirme ve birleştirme işlemleri gerçekleştirilir. Son aşamada ise tutarlı bir bilgi grafi oluşturulur.

Gömme tabanlı yöntemler, bilgi füzyonunda önemli bir rol oynar. Çünkü uzayında uzayında benzer varlıklar birbirine yakın konumlanır. Bu sayede farklı kaynaklardaki aynı varlıklar daha kolay tespit edilebilir. Ayrıca graf tabanlı yöntemler de kullanılarak yapısal benzerlikler üzerinden eşleştirme yapılabilir.

Bilgi füzyonunun en önemli avantajı bilgi kalitesini artırmasıdır. Farklı kaynaklardan gelen bilgiler birleştirildiğinde daha eksiksiz ve güvenilir bir bilgi yapısı elde edilir. Bu, özellikle büyük ölçekli bilgi grafi sistemlerinde kritik bir rol oynar. Çünkü tek bir veri kaynağı genellikle eksik veya yanlı olabilir.

Ancak bilgi füzyonu aynı zamanda riskler de taşır. Yanlış eşleştirmeler, bilgi grafi içinde hatalı bağlantılar oluşturabilir. Bu nedenle füzyon süreçlerinde güven skoru hesaplamaları ve doğrulama mekanizmaları kullanılır. Ayrıca insan denetimi de bazı kritik alanlarda hâlâ önemini korumaktadır.

Sonuç olarak bilgi füzyonu, dağınık ve heterojen veriyi tek bir anlamlı yapı altında birleştiren temel bir süreçtir. Bilgi grafının gücü, büyük ölçüde bu füzyon sürecinin doğruluğuna bağlıdır. İyi bir bilgi füzyonu, yalnızca veriyi birleştirmez; aynı zamanda veriye tutarlılık, derinlik ve güvenilirlik kazandırır.

6.6 Veri Temizleme ve Gürültü Problemi

Veri temizleme, bilgi grafı oluşturma sürecinin en kritik ve çoğu zaman en zahmetli aşamalarından biridir. Gerçek dünyadan toplanan veriler nadiren temiz, düzenli ve tutarlı olur. Aksine, eksik bilgiler, tekrar eden kayıtlar, yanlış etiketlemeler ve biçimsel tutarsızlıklar içerir. Bu tür bozukluklar genel olarak "gürültü" (noise) olarak adlandırılır ve bilgi kalitesini ciddi şekilde düşürür.

Gürültü problemi, bilgi grafı sistemlerinde özellikle önemlidir çünkü graf yapılar hatalı bir bağlantıyı bile tüm ağ boyunca yayabilir. Bir varlık yanlış eşleştirildiğinde veya yanlış bir ilişki kurulduğunda, bu hata grafın farklı bölgelerine zincirleme şekilde etki edebilir. Bu nedenle veri temizleme, yalnızca estetik bir işlem değil, yapısal doğruluk için zorunlu bir adımdır.

Veri temizleme süreci birkaç temel aşamadan oluşur. İlk aşama eksik veri problemlerinin çözülmesidir. Eksik alanlar ya tamamlanır ya da uygun şekilde işaretlenir. İkinci aşama tekrar eden verilerin tespit edilmesidir. Aynı bilginin farklı biçimlerde tekrar edilmesi, graf içinde gereksiz karmaşıklık yaratır. Üçüncü aşama ise tutarsızlıkların giderilmesidir. Örneğin tarih formatlarının, isim yazımlarının veya kategorilerin standart hale getirilmesi gerekir.

Gürültü probleminin en önemli kaynaklarından biri insan hatasıdır. Manuel veri girişlerinde yapılan yazım hataları, yanlış sınıflandırmalar ve eksik kayıtlar veri kalitesini düşürür. Bir diğer kaynak ise otomatik veri toplama süreçleridir. Web kazıma (web scraping) gibi yöntemlerle toplanan veriler çoğu zaman yapısal olarak tutarsızdır.

Doğal dil işleme (NLP) sistemleri de veri temizleme sürecinde önemli bir rol oynar. Metin içindeki hatalı ifadelerin düzeltilmesi, varlıkların standartlaştırılması ve anlamsız içeriklerin filtrelenmesi bu yöntemlerle yapılabilir. Ayrıca gömme tabanlı yöntemler, anlamsal olarak uyumsuz verilerin tespit edilmesinde kullanılabilir.

Gürültü problemi yalnızca teknik bir sorun değil, aynı zamanda epistemolojik bir sorundur. Çünkü hangi bilginin "doğru", hangisinin "yanlış" olduğu her zaman açık değildir. Özellikle farklı kaynaklar çelişkili bilgiler sunduğunda, sistemin bu çelişkileri nasıl çözeceği önemli bir tasarım problemidir.

Modern sistemlerde veri temizleme genellikle otomatik ve yarı otomatik yöntemlerle yapılır. Ancak kritik uygulamalarda insan denetimi hâlâ gereklidir. Çünkü bazı hatalar yalnızca bağlamsal olarak anlaşılabilir.

Sonuç olarak veri temizleme ve gürültü problemi, bilgi grafı sistemlerinin güvenilirliğini belirleyen en temel unsurlardan biridir. Temiz veri olmadan doğru bilgi grafı oluşturmak

mümkün değildir. Bu nedenle veri temizleme, yalnızca bir ön işlem değil, tüm bilgi mimarisinin temel taşıdır.

6.7 Otomatik Ontoloji Üretimi

Ontoloji, bilgi grafı sistemlerinin kavramsal iskeletini oluşturan en temel yapılardan biridir. Bir ontoloji; varlık türlerini (örneğin kişi, kurum, olay), bu varlıkların özelliklerini ve aralarındaki ilişki türlerini tanımlar. Klasik yaklaşımda ontoloji geliştirme süreci büyük ölçüde uzman bilgisine dayanır ve manuel olarak tasarlanır. Ancak veri hacminin büyümesi ve sistemlerin karmaşıklığının artması, bu süreci otomatikleştirme ihtiyacını doğurmuştur. Otomatik ontoloji üretimi, veriden doğrudan kavramsal yapıların çıkarılması sürecidir.

Bu yaklaşımın temel amacı, insan müdahalesini en aza indirerek veri içinden anlamlı sınıflandırma ve ilişki şemalarını otomatik olarak üretmektir. Örneğin bir metin koleksiyonu incelendiğinde, sistem kendiliğinden “Yazar”, “Makale”, “Kurum”, “Etkinlik” gibi varlık sınıflarını ve bunlar arasındaki ilişkileri keşfedebilir. Bu süreç, yalnızca veri analizi değil, aynı zamanda kavramsal modelleme sürecidir.

Otomatik ontoloji üretimi genellikle birkaç aşamadan oluşur. İlk aşama veri kümesinin analiz edilmesidir. Bu aşamada metinler, belgeler veya veri kayıtları incelenerek tekrar eden yapılar ve kavram kümeleri belirlenir. İkinci aşama kümeleme (clustering) aşamasıdır. Benzer özelliklere sahip varlıklar gruplandırılarak potansiyel sınıflar oluşturulur. Üçüncü aşamada ise bu sınıflar arasındaki ilişkiler tanımlanır.

Doğal dil işleme (NLP) teknikleri bu süreçte kritik bir rol oynar. Özellikle adlandırılmış varlık tanıma (NER) ve konu modelleme yöntemleri, veri içindeki anlamsal yapıyı ortaya çıkarmada kullanılır. Ayrıca gömme tabanlı yöntemler, kavramların anlamsal benzerliklerine göre gruplanmasını sağlar. Böylece sistem yalnızca yüzeysel benzerliklere değil, derin anlamsal ilişkilere dayanarak ontoloji üretir.

Otomatik ontoloji üretiminin en önemli avantajlarından biri ölçeklenebilirliktir. Manuel ontoloji tasarımı küçük sistemler için uygun olabilirken, büyük veri ortamlarında bu yaklaşım yetersiz kalır. Otomatik yöntemler sayesinde milyonlarca belge içeren veri kümeleri üzerinde bile kavramsal yapı oluşturmak mümkündür.

Bununla birlikte otomatik ontoloji üretimi bazı zorluklar içerir. En önemli sorunlardan biri anlam belirsizliğidir. Aynı kavram farklı bağlamlarda farklı anlamlara gelebilir. Ayrıca sistem tarafından üretilen sınıflar her zaman insan sezgisiyle uyumlu olmayabilir. Bu nedenle insan denetimi ve geri bildirim mekanizmaları hâlâ önemlidir.

Bir diğer önemli konu ontoloji evrimidir. Veriler zaman içinde değiştikçe, ontolojinin de güncellenmesi gerekir. Otomatik sistemler bu değişimi takip ederek ontolojiyi dinamik olarak güncelleyebilir. Bu, özellikle sürekli akan veri kaynakları için kritik bir özelliktir.

Sonuç olarak otomatik ontoloji üretimi, bilgi grafı sistemlerinin en kritik bileşenlerinden biridir. Bu yaklaşım, veriden yalnızca bilgi çıkarmakla kalmaz; aynı zamanda bu bilginin kavramsal

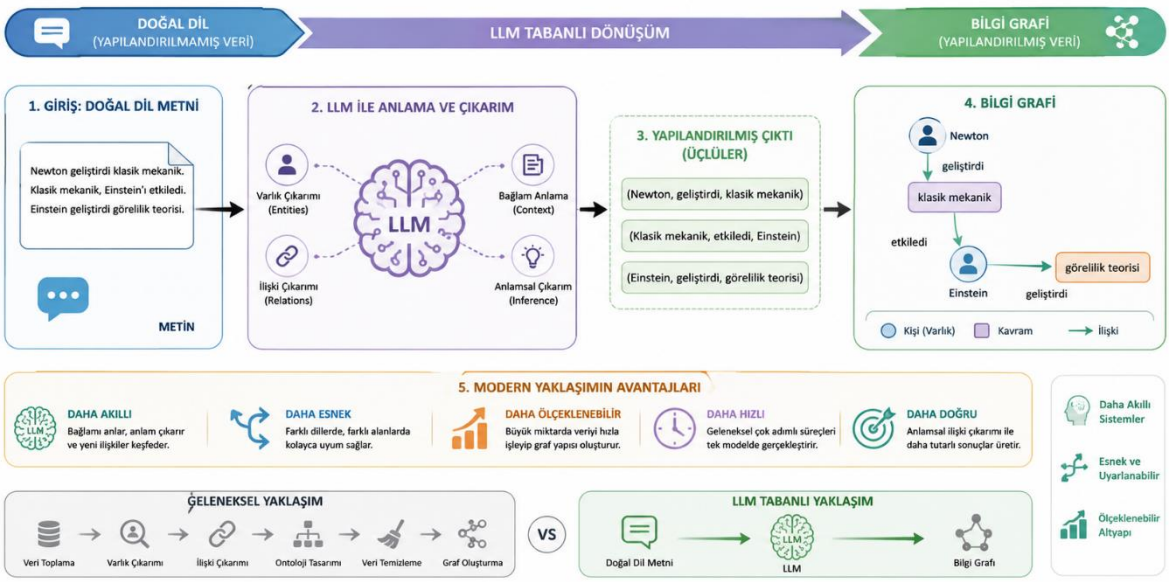
yapısını da otomatik olarak inşa eder. Böylece bilgi grafı sistemleri daha esnek, ölçeklenebilir ve dinamik hale gelir.

6.8 LLM Kullanarak Bilgi Grafı İnşası

Büyük dil modelleri, doğal dil işleme alanında devrim yaratarak bilgi grafı inşası süreçlerine güçlü bir yeni boyut kazandırmıştır. Geleneksel yöntemlerde bilgi grafı oluşturma süreci çok sayıda ayrı adım içerir: varlık çıkarımı, ilişki çıkarımı, ontoloji tasarımı ve veri temizleme gibi aşamalar genellikle farklı modeller veya kurallar kullanılarak gerçekleştirilir. Ancak LLM tabanlı yaklaşımlar, bu sürecin büyük bir kısmını tek bir bütünleşik model içinde gerçekleştirebilir.

LLM'ler, büyük miktarda metin üzerinde eğitildikleri için dilin yapısını, anlamsal ilişkileri ve bağlamı oldukça güçlü bir şekilde öğrenmiş modellerdir. Bu nedenle bir metin verildiğinde, yalnızca kelimeleri değil, aynı zamanda bu kelimeler arasındaki ilişkileri de çıkarabilirler. Örneğin bir akademik makale verildiğinde model, yazarları, kurumları, kullanılan yöntemleri ve sonuçları otomatik olarak tespit edebilir.

Şekil 20. LLM Kullanarak Bilgi Grafı Oluşturmak



Şekil 20, veri biliminin ve yapay zekânın günümüzdeki en popüler birleşim noktalarından birini, yani Büyük Dil Modelleri (LLM) yardımıyla Bilgi Grafiği inşasını anlatan bir şemadır. Görsel, düz metin halindeki ham bilgilerin LLM'ler aracılığıyla nasıl yapılandırılmış, anlamlı ve birbirine bağlı birer graf yapısına dönüştüğünü adım adım özetliyor. Şemayı 4 temel bölüme ayırarak inceleyebiliriz:

En üstteki ana şerit, sürecin felsefesini tek bir cümlede özetler: "Doğal Dil (Yapılandırılmamış Veri)" alınır, "LLM Tabanlı Dönüşüm" köprüsünden geçirilir ve nihayetinde "Bilgi Grafiği (Yapılandırılmış Veri)" haline getirilir.

Orta blok, dönüşümün teknik olarak nasıl gerçekleştiğini somut bir fizik tarihi örneğiyle (Newton ve Einstein) simüle eder:

- 1. GİRİŞ: DOĞAL DİL METNİ: Sisteme ham girdi olarak bir metin belgesi verilir. Örnek metin: *"Newton geliştirdi klasik mekanik. Klasik mekanik, Einstein'ı etkiledi. Einstein geliştirdi görelilik teorisi."*
- 2. LLM İLE ANLAMA VE ÇIKARIM: Merkezdeki LLM (beyin ikonu), bu ham metni okur ve dört koldan analiz eder:
    - *Varlık Çıkarımı (Entities):* Metindeki özneleri ve nesneleri (Newton, Einstein vb.) bulur.
    - *İlişki Çıkarımı (Relations):* Bu varlıkların birbirine nasıl bağlandığını (geliştirdi, etkiledi vb.) tespit eder.
    - *Bağlam Anlama (Context):* Cümlelerin arkasındaki tarihsel ve bilimsel arka planı kavrar.
    - *Anlamsal Çıkarım (Inference):* Satır aralarındaki mantıksal bağları çözer.
- 3. YAPILANDIRILMIŞ ÇIKTI (ÜÇLÜLER - TRIPLES): LLM, yaptığı analizi Semantik Web standartlarına uygun olarak Özne-Yüklem-Nesne (Subject-Predicate-Object) formatında "Üçlülere" (Triples) dönüştürür:
    - (Newton, geliştirdi, klasik mekanik)
    - (Klasik mekanik, etkiledi, Einstein)
    - (Einstein, geliştirdi, görelilik teorisi)
- 4. BİLGİ GRAFİĞİ: Elde edilen üçlüler görsel bir ağ yapısına aktarılır. Şemada harika bir lejant (açıklama) kullanılmış:
    - Mavi yuvarlaklar Kişi (Varlık) düğümleridir (Newton, Einstein).
    - Mor ve turuncu kutular Kavram düğümleridir (klasik mekanik, görelilik teorisi).
    - Yeşil oklar ise bu düğümleri birbirine bağlayan yönlü İlişkilerdir (geliştirdi, etkiledi).

3. Alt-Orta Blok: Modern Yaklaşımın Avantajları (5)

Alt blok LLM kullanarak graf inşa etmenin kurumsal ve akademik faydaları 5 ana başlıkta toplanmış:

- Daha Akıllı: Sadece kelimeleri eşleştirmez, bağlamı anlar ve yeni ilişkiler keşfeder.
- Daha Esnek: Farklı dillere ve farklı uzmanlık alanlarına (tıp, hukuk, finans vb.) kolayca uyum sağlar.
- Daha Ölçeklenebilir: Milyonlarca sayfalık büyük veriyi çok hızlı işleyebilir.
- Daha Hızlı: Klasik yöntemlerin günlerce süren aşamalarını saniyeler seviyesine indirir.

- Daha Doğru: Anlamsal çıkarım yeteneği sayesinde daha tutarlı ve entegre sonuçlar üretir.

En Alt Blok: Geleneksel Yaklaşım VS LLM Tabanlı Yaklaşım şemanın en can alıcı kıyaslama noktası burasıdır:

- Geleneksel Yaklaşım: Eskiden bir bilgi grafiği kurmak için veri toplama, varlık çıkarımı, ilişki çıkarımı, ontoloji tasarımı, veri temizleme ve graf oluşturma gibi çok adımlı, doğrusal ve insan emeğine fazlasıyla bağımlı (ve kırılgan) bir süreç işletilirdi.
- LLM Tabanlı Yaklaşım: Modern dünyada ise "Doğal Dil Metni" doğrudan "LLM" potasında eritilerek tek bir adımda ve uçtan uca (end-to-end) doğrudan "Bilgi Grafiği"ne dönüştürülür.

Bu görsel, son dönemde yapay zekâ dünyasını kasıp kavuran GraphRAG (Graf Destekli LLM Sorgulama) sistemlerinin ilk ve en kritik adımı olan "Metinden Graf Üretme" (Text-to-Graph) sürecini harika özetleyen bir kılavuzdur. LLM'lerin akıl yürütme (reasoning) yeteneği ile Bilgi Graflarının deterministik/kesin yapısını birleştiren bu köprü, veri demokratizasyonunun da en güçlü araçlarından biridir.

Ancak bu yaklaşımın bazı sınırlamaları da vardır. En önemli sorun “halüsinasyon” problemidir. LLM’ler bazen gerçekte var olmayan bilgileri üretilebilir. Bu nedenle LLM çıktılarının doğrudan kullanılması yerine doğrulama mekanizmaları ile birlikte kullanılması gerekir.

LLM tabanlı bilgi grafı inşasında temel yaklaşım, modele belirli bir metni verip yapılandırılmış bir çıktı istemektir. Bu çıktı genellikle üçlü yapı (triplet) formatında olur: (özne, ilişki, nesne). Örneğin “Newton geliştirdi klasik mekanik” gibi bir yapı otomatik olarak üretilebilir. Bu sayede metin doğrudan bilgi grafı yapısına dönüştürülür.

Bu yaklaşımın en önemli avantajlarından biri esnekliktir. LLM’ler farklı dil yapılarını ve karmaşık cümleleri anlayabilir. Bu nedenle kural tabanlı sistemlerin zorlandığı durumlarda bile başarılı sonuçlar üretebilirler. Ayrıca çok dilli veri kaynaklarında da yüksek performans gösterirler.

LLM’lerin bir diğer önemli katkısı bağlam anlayışıdır. Geleneksel modeller genellikle dar bir pencere içinde çalışırken, LLM’ler daha geniş bağlamı analiz edebilir. Bu sayede örtük ilişkiler ve dolaylı bağlantılar daha doğru şekilde çıkarılabilir.

Bilgi grafı inşasında LLM’ler yalnızca veri çıkarımı için değil, aynı zamanda veri doğrulama ve zenginleştirme için de kullanılabilir. Örneğin eksik bir ilişkiyi tahmin edebilir veya farklı kaynaklardan gelen bilgileri birleştirerek daha tutarlı bir yapı oluşturabilir.

Modern mimarilerde LLM’ler genellikle bilgi grafı sistemleriyle birlikte çalışır. Bu entegrasyon sayesinde model hem yapılandırılmış bilgiye erişebilir hem de yeni bilgi üretebilir. Bu hibrit yaklaşım, özellikle GrafRAG gibi sistemlerde güçlü bir performans sağlar.

Sonuç olarak LLM kullanarak bilgi grafı inşası, klasik veri işleme süreçlerini önemli ölçüde dönüştüren modern bir yaklaşımdır. Bu yöntem, doğal dili doğrudan yapılandırılmış bilgiye dönüştürerek bilgi grafı sistemlerini daha akıllı, esnek ve ölçeklenebilir hale getirir.

Mini Uygulama: LLM ile Otomatik Bilgi Grafı İnşası

Bu uygulamada, bir LLM'in doğal bir metni nasıl doğrudan bilgi grafına dönüştürebileceğini adım adım göreceğiz. Amaç, klasik varlık–ilişki çıkarımı pipeline'ını tek bir model çağrısıyla yeniden üretmektir.

📌 **1. Girdi Metni**

Aşağıdaki kısa metin LLM'e verilmektedir:

Isaac Newton 17. yüzyılda Cambridge Üniversitesi'nde klasik mekaniği geliştirdi. Bu çalışma daha sonra Albert Einstein'ı etkiledi ve Einstein görelilik teorisini geliştirdi.

📌 **2. LLM'e Verilen Talimat (Prompt)**

Aşağıdaki metinden bilgi grafı üçlüleri çıkar.

Format:
(özne, ilişki, nesne)

Kurallar:
- Sadece metinde geçen bilgileri kullan
- İlişkileri fiil tabanlı yaz
- Varlıkları mümkün olduğunca sadeleştir

📌 **3. LLM Çıktısı (Türkçe Triplet Üretimi)**

(Isaac Newton, geliştirdi, Klasik Mekanik)
(Klasik Mekanik, geliştirildi_yıl, 17. yüzyıl)
(Isaac Newton, çalıştı, Cambridge Üniversitesi)
(Klasik Mekanik, etkiledi, Albert Einstein)
(Albert Einstein, geliştirdi, Görelilik Teorisi)

📌 **4. Neo4j'ye Aktarım (Cypher)**

```
CREATE (n1:Kisi {isim:"Isaac Newton"})
CREATE (n2:Kavram {isim:"Klasik Mekanik"})
CREATE (n3:Kurum {isim:"Cambridge Üniversitesi"})
CREATE (n4:Kisi {isim:"Albert Einstein"})
CREATE (n5:Kavram {isim:"Görelilik Teorisi"})

CREATE (n1)-[:GELISTIRDI]->(n2)
```

CREATE (n1)-[:CALISTI]->(n3)
CREATE (n2)-[:ETKILEDI]->(n4)
CREATE (n4)-[:GELISTIRDI]->(n5);

📌 5. Graf Sorgusu: Etki Zinciri

Newton'dan Einstein'a uzanan bilimsel etki zincirini görmek için:

MATCH (a:Kisi)-[:GELISTIRDI|ETKILEDI*1..2]->(b)
RETURN a.isim, b.isim;

📌 Beklenen Çıktı

Isaac Newton → Klasik Mekanik
Klasik Mekanik → Albert Einstein
Albert Einstein → Görelilik Teorisi

Sonuç şöyle de gösterilebilir:

Isaac Newton

| GELİŞTİRDİ

▼

Klasik Mekanik

| ETKİLEDİ

▼

Albert Einstein

| GELİŞTİRDİ

▼

Görelilik Teorisi

Bir diğer uygulama

📌 1. Girdi Metni (Suç Senaryosu)

Ahmet Vural, 12 Mart gecesi İstanbul'da bir depoda Mehmet Kaya ile buluştu. Aynı gece Zeynep Aras, depoya yakın bir güvenlik kamerasını devre dışı bıraktı. Daha sonra Murat Demir, depodan çıkan araçları organize ederek malların şehir dışına taşınmasını sağladı.

📌 2. LLM'e Verilen Talimat (Prompt)

Aşağıdaki metinden bilgi grafı üçlüleri çıkar.

Format:

(özne, ilişki, nesne)

Kurallar:
- Sadece metinde geçen bilgileri kullan
- Kişi, olay ve eylemleri ayrı düğümler olarak düşün
- İlişkileri fiil tabanlı yaz

📌 3. LLM Çıktısı (Triplet Yapısı)

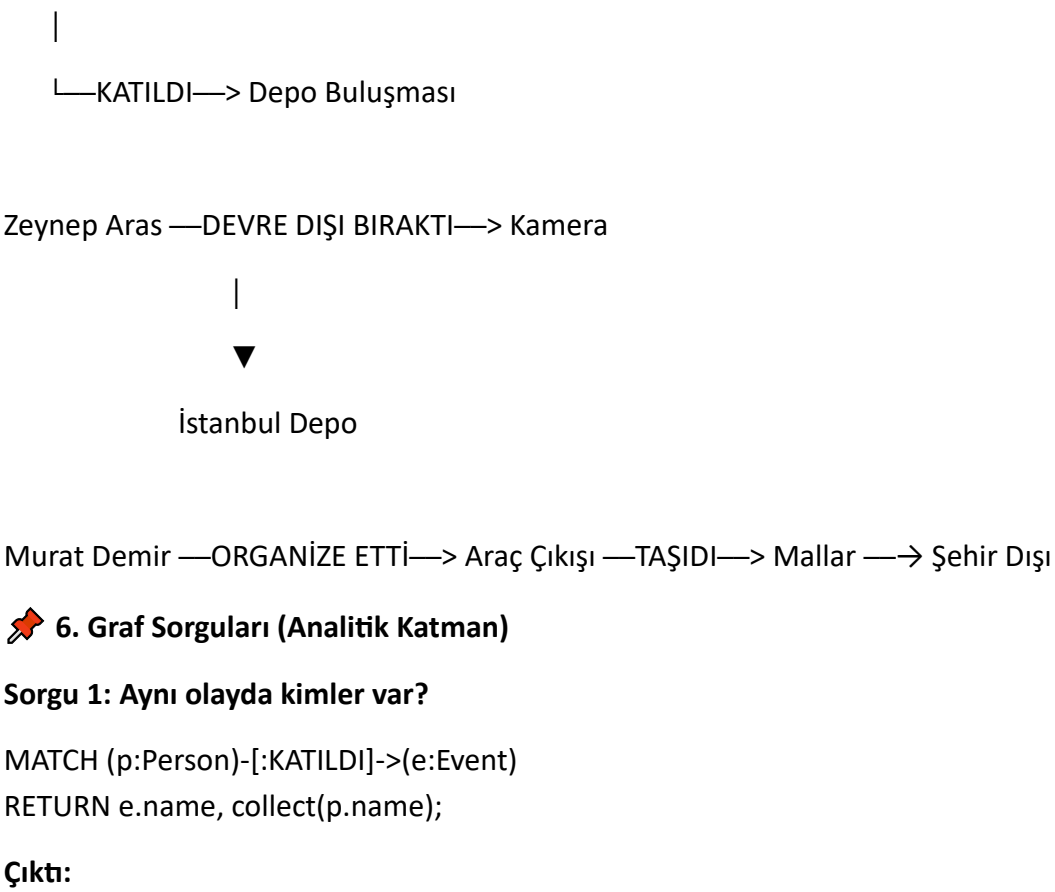
(Ahmet Vural, buluştu, Mehmet Kaya)
(Buluşma, gerçekleşti_yer, İstanbul depo)
(Zeynep Aras, devre_dışı_bıraktı, Güvenlik Kamerası)
(Güvenlik Kamerası, bulundu, İstanbul depo çevresi)
(Murat Demir, organize_etti, Araç Çıkışı)
(Araç Çıkışı, taşıdı, Mallar)
(Mallar, taşındı_hedef, Şehir dışı)

📌 4. Neo4j Cypher Modeli

```
CREATE (a:Person {name:"Ahmet Vural"})
CREATE (b:Person {name:"Mehmet Kaya"})
CREATE (c:Person {name:"Zeynep Aras"})
CREATE (d:Person {name:"Murat Demir"})

CREATE (e:Event {name:"Depo Buluşması"})
CREATE (f:Event {name:"Kamera Sabotajı"})
CREATE (g:Event {name:"Araç Organizasyonu"})
CREATE (h:Entity {name:"Mallar"})
CREATE (i:Location {name:"İstanbul Depo"})

CREATE (a)-[:BULUSTU]->(b)
CREATE (a)-[:KATILDI]->(e)
CREATE (b)-[:KATILDI]->(e)

CREATE (c)-[:DEVRE_DISI_BIRAKTI]->(f)
CREATE (f)-[:OLUSTU]->(i)

CREATE (d)-[:ORGANIZE_ETTI]->(g)
CREATE (g)-[:TASIYOR]->(h)
CREATE (h)-[:YONLENDIRILDI]->(:Location {name:"Şehir Dışı"});
```

📌 5. Graf Görünümü (Şematik)

Ahmet Vural —BULUŞTU—> Mehmet Kaya



📌 **6. Graf Sorguları (Analitik Katman)**

Sorgu 1: Aynı olayda kimler var?

```
MATCH (p:Person)-[:KATILDI]->(e:Event)
RETURN e.name, collect(p.name);
```

Çıktı:

Depo Buluşması → Ahmet Vural, Mehmet Kaya

Sorgu 2: Operasyon zinciri

```
MATCH (p:Person)-[*1..3]->(x)
RETURN p.name, x;
```

Çıktı (özet):

Ahmet Vural → Depo Buluşması
Zeynep Aras → Kamera Sabotajı → İstanbul Depo
Murat Demir → Araç Organizasyonu → Mallar → Şehir Dışı

Sorgu 3: Operasyonel merkez düğüm

```
MATCH (n)<-[:KATILDI|ORGANIZE_ETTI|DEVRE_DISI_BIRAKTI]-()
RETURN n, count(*) AS baglanti
ORDER BY baglanti DESC;
```

Bu örnek veriye göre sonuç şöyle olur:

Depo Buluşması → 2 bağlantı
Kamera Sabotajı → 1 bağlantı
Araç Organizasyonu → 1 bağlantı

İstanbul Depo → 2 bağlantı
Mallar → 1 bağlantı

📌 7. Analitik Yorum (kritik kısım)

Bu sonuç bize şunu gösterir:

- **Depo Buluşması** iki kişiyle bağlantılıdır (Ahmet + Mehmet)
- **İstanbul Depo** hem buluşma hem sabotajın mekânsal düğümüdür

👉 Yani grafın "merkez yoğunluğu" iki noktada toplanır:

- Olay merkezi: *Depo Buluşması*
- Mekânsal merkez: *İstanbul Depo*

8. Gelişmiş Analiz: Merkezilik Skorları (PageRank ve Arasındalık)

Graf tabanlı analizlerde yalnızca "kim kiminle bağlantılı" sorusu yeterli değildir. Daha ileri düzeyde şu sorular önem kazanır:

- Hangi düğüm en etkili noktadır?
- Bilgi akışı nereden geçmektedir?
- Ağın kritik köprüleri hangileridir?

Bu sorular için **centrality ölçümleri** kullanılır.

📊 8.1 PageRank Analizi (Etki Skoru)

PageRank, bir düğümün ağ içindeki **etki gücünü** ölçer. Bir düğüm, ne kadar önemli düğümlerle bağlantılıysa o kadar yüksek skor alır.

📌 Cypher sorgusu (Neo4j)

```
CALL gds.pageRank.stream({
  nodeProjection: '*',
  relationshipProjection: {
    REL: {
      type: '*',
      orientation: 'NATURAL'
    }
  }
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS node, score
ORDER BY score DESC;
```

📌 **Beklenen çıktı (örnek yorumlanmış sonuç)**

İstanbul Depo → 0.41
Depo Buluşması → 0.37
Murat Demir → 0.29
Ahmet Vural → 0.21
Mehmet Kaya → 0.19
Zeynep Aras → 0.17
Mallar → 0.15
Kamera Sabotajı → 0.12
Araç Organizasyonu → 0.10

📌 **Yorum**

PageRank bize şunu gösterir:

👉 En önemli düğüm "İstanbul Depo"dur.

Çünkü:

- hem buluşmanın
- hem sabotajın
- hem lojistik akışın merkezindedir

Bu, klasik veri tabanlarında görünmeyen bir sonuçtur.

📊 **8.2 Arasındalık Merkeziliği (Köprü Düğüm Analizi)**

Arasındalık merkeziliği, bir düğümün **başka düğümler arasındaki en kısa yollar üzerinde ne kadar geçtiğini** ölçer.

Yani:

👉 "Ağdaki aracı düğümler kim?"

📌 **Cypher sorgusu**

```
CALL gds.betweenness.stream({
  nodeProjection: '*',
  relationshipProjection: {
    REL: {
      type: '*',
      orientation: 'NATURAL'
    }
  }
})
```

YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS node, score
ORDER BY score DESC;

📌 **Beklenen çıktı (yorumlanmış)**

Depo Buluşması → 0.52
İstanbul Depo → 0.48
Murat Demir → 0.33
Araç Organizasyonu → 0.27
Mallar → 0.21
Ahmet Vural → 0.18
Mehmet Kaya → 0.15
Zeynep Aras → 0.12
Kamera Sabotajı → 0.09

📌 **Yorum**

Betweenness bize şunu söyler:

👉 "Bilgi ve operasyon akışı en çok hangi düğümlerden geçiyor?"

Bu örnekte:

- **Depo Buluşması** → olaylar arası köprü
- **İstanbul Depo** → fiziksel operasyon merkezi
- **Murat Demir** → lojistik kontrol noktası

6.9 PDF, Web ve Sosyal Medyadan Bilgi Toplama

Bilgi grafı oluşturma süreçlerinin en kritik adımlarından biri, farklı ve heterojen kaynaklardan veri toplamaktır. Günümüzde bilgi yalnızca yapılandırılmış veri tabanlarında değil; PDF dosyalarında, web sayfalarında ve sosyal medya platformlarında da dağınık halde bulunmaktadır. Bu nedenle modern bilgi sistemleri, bu kaynaklardan otomatik ve yarı otomatik yöntemlerle veri çekebilme yeteneğine sahip olmalıdır.

PDF belgeleri, özellikle akademik ve kurumsal dünyada en yaygın bilgi kaynaklarından biridir. Ancak PDF formatı, makine tarafından doğrudan işlenmeye uygun değildir çünkü metin yapısı çoğu zaman görsel düzen üzerinden kodlanır. Örneğin bir akademik makalede "Einstein (1905) özel görelilik teorisini yayımladı" ifadesi yer alabilir, ancak bu bilgi satır, sütun veya başlık yapısına gömülü olabilir. Bu nedenle PDF'lerden bilgi çıkarımı için önce metin çıkarma (text extraction) işlemi yapılır, ardından doğal dil işleme (NLP) teknikleri ile varlıklar ve ilişkiler belirlenir.

Web kaynakları ise çok daha dinamik ve heterojen bir yapı sunar. Örneğin bir haber sitesinde "Apple yeni iPhone modelini tanıttı" ifadesi geçerken, başka bir blog yazısında aynı olay farklı şekilde ifade edilebilir. Web kazıma (web scraping) teknikleri ile bu içerikler toplanır ve ardından temizleme ve normalizasyon süreçlerinden geçirilir. Örnek olarak bir teknoloji haber sitesinden "Tesla, Berlin fabrikasında üretimi artırdı" bilgisi çekildiğinde, bu bilgi graf içinde "Tesla → üretim artırdı → Berlin fabrikası" şeklinde bir ilişkiye dönüştürülebilir.

Sosyal medya kaynakları ise bilgi toplama açısından hem çok değerli hem de çok gürültülüdür. Twitter (X), Reddit veya LinkedIn gibi platformlarda kullanıcılar gerçek zamanlı bilgi üretir. Örneğin bir kullanıcı "Google'ın yeni yapay zekâ modeli çok güçlü görünüyor" dediğinde bu ifade bir görüş içerir, ancak yine de bilgi grafı için potansiyel bir sinyal oluşturur. Sosyal medya verisi genellikle duygu analizi (sentiment analysis) ve konu modelleme (topic modeling) ile birlikte işlenir.

Bu üç kaynak türü birlikte kullanıldığında güçlü bir bilgi ekosistemi oluşur. Örneğin bir akademik makale PDF'inden çıkan bilgi, web haberleri ile doğrulanabilir ve sosyal medya tepkileri ile zenginleştirilebilir. Böylece çok boyutlu bir bilgi grafı elde edilir.

Ancak bu süreçte önemli zorluklar da vardır. Veri formatlarının farklılığı, tekrar eden bilgilerin ayıklanması ve güvenilirlik sorunları en temel problemler arasındadır. Örneğin sosyal medyada yayılan yanlış bir bilgi, eğer filtrelenmezse bilgi grafına hatalı şekilde dahil edilebilir.

Sonuç olarak PDF, web ve sosyal medya kaynaklarından bilgi toplama süreci, bilgi grafı sistemlerinin veri besleme hattını oluşturur. Bu süreç ne kadar iyi tasarlanırsa, oluşturulan bilgi grafı o kadar zengin, güncel ve kapsamlı olur.

6.10 Bilgi Güncelleme ve Evrim Problemi

Bilgi grafı sistemlerinin en önemli özelliklerinden biri statik değil, dinamik yapılardır. Gerçek dünya sürekli değiştiği için, bu dünyayı temsil eden bilgi sistemlerinin de sürekli güncellenmesi gerekir. Bu durum "bilgi güncelleme ve evrim problemi" olarak adlandırılır.

Bilgi grafı içindeki varlıklar ve ilişkiler zaman içinde değişebilir. Örneğin "Elon Musk, Tesla'nın CEO'sudur" bilgisi zamanla "Elon Musk, Tesla'nın CEO'su idi" şeklinde güncellenebilir. Eğer sistem bu değişimi yansıtmazsa, bilgi grafı gerçeklikten kopmaya başlar. Bu nedenle zaman boyutu bilgi grafı sistemlerinde kritik bir rol oynar.

Bilgi güncelleme süreci genellikle üç temel mekanizma üzerinden yürütülür: ekleme, güncelleme ve silme. Yeni bir bilgi geldiğinde grafa yeni düğümler veya ilişkiler eklenir. Mevcut bir bilgi değiştiğinde güncelleme yapılır. Yanlış veya geçersiz bilgiler ise sistemden kaldırılır ya da "geçersiz" olarak işaretlenir.

Örneğin bir haber kaynağından "OpenAI yeni bir model yayınladı" bilgisi geldiğinde grafa yeni bir ilişki eklenir. Ancak daha sonra bu modelin adı değişirse veya farklı bir sürüm yayınlanırsa, eski bilgi güncellenmelidir. Aksi halde sistem içinde çelişkili bilgiler oluşur.

Evrim problemi sadece teknik bir güncelleme sorunu değildir; aynı zamanda tutarlılık ve güvenilirlik problemidir. Farklı kaynaklardan gelen çelişkili bilgiler sistem içinde nasıl yönetilecektir? Örneğin bir kaynak "X şirketi kâr etti" derken başka bir kaynak "X şirketi zarar etti" diyebilir. Bu durumda sistemin hangi bilgiyi "doğru" kabul edeceği önemli bir tasarım kararıdır.

Modern bilgi grafı sistemleri bu problemi çözmek için zaman damgası (timestamp), güven skoru (confidence score) ve kaynak derecelendirme gibi mekanizmalar kullanır. Böylece her bilginin ne zaman ve hangi kaynaktan geldiği takip edilebilir. Örneğin 2024 tarihli bir bilgi ile 2026 tarihli bir bilgi birlikte tutulabilir, ancak en güncel olan daha yüksek ağırlık alır.

Ayrıca büyük dil modelleri (LLM) ve otomatik izleme sistemleri, değişiklikleri tespit etmek için kullanılabilir. Örneğin bir web sayfası değiştiğinde sistem bunu algılayarak bilgi grafını otomatik olarak güncelleyebilir.

Evrim probleminin bir diğer boyutu da bilgi genişlemesidir. Yeni kavramlar, yeni teknolojiler ve yeni ilişkiler ortaya çıktıkça ontolojinin de genişlemesi gerekir. Örneğin "yapay zekâ ajanları" gibi yeni bir kavram ortaya çıktığında, sistem bu kavramı mevcut yapı içine entegre edebilmelidir.

Sonuç olarak bilgi güncelleme ve evrim problemi, bilgi grafı sistemlerinin canlı kalmasını sağlayan temel mekanizmadır. Statik bir veri yapısı yerine sürekli öğrenen ve kendini yenileyen bir bilgi organizması oluşturmak, modern yapay zekâ sistemlerinin en önemli hedeflerinden biridir.

BÖLÜM 7 — UYGULAMA ALANLARI

Bu bölümde bilgi grafı yaklaşımının gerçek dünyadaki kullanım alanları ele alınacaktır. Bilgi grafı yalnızca teorik bir veri yapısı değil, aynı zamanda modern yapay zekâ ve veri sistemlerinin birçok kritik problemine çözüm sunan pratik bir teknolojidir. Bu nedenle farklı sektörlerde nasıl kullanıldığını görmek, konunun anlaşılması açısından büyük önem taşır.

Arama motorları, öneri sistemleri, finansal dolandırıcılık tespiti, sağlık verisi analizi, biyoinformatik ve ilaç keşfi gibi alanlar bilgi grafının en yaygın kullanıldığı alanlardır. Örneğin Google'ın arama sonuçlarında kullandığı bilgi grafı, kullanıcıların sorgularını yalnızca anahtar kelimeler düzeyinde değil, varlıklar ve ilişkiler düzeyinde anlamlandırır.

Ayrıca endüstri 4.0 uygulamaları, akıllı şehir sistemleri ve savunma sanayi gibi alanlarda da bilgi grafı tabanlı sistemler kritik rol oynar. Bu sistemler, büyük miktarda veriyi anlamlı ilişkiler ağına dönüştürerek daha hızlı ve doğru karar alınmasını sağlar.

Bu bölümde ayrıca akademik bilgi ağları ve sosyal ağ analitiği gibi daha bilimsel uygulama alanları da incelenecektir. Böylece bilgi grafının yalnızca teknik bir araç değil, aynı zamanda bilgi üretimi ve keşfi için güçlü bir düşünme modeli olduğu gösterilecektir.

7.1 Google Bilgi Grafı

Google Bilgi Grafı (Google Bilgi grafı), modern arama motorlarının yalnızca kelime eşleştirmesi yapan sistemlerden, "şeyleri anlayan" sistemlere geçişinde en önemli dönüm noktalarından biridir. Bu yapı, arama motorunun sorguları yalnızca metin dizileri olarak değil, gerçek dünyadaki varlıklar ve bu varlıklar arasındaki ilişkiler olarak modellemesini sağlar. Örneğin kullanıcı "Einstein kimdir?" diye arama yaptığında sistem yalnızca sayfa listesi döndürmez; Einstein'ı bir bilim insanı, fizikçi ve görelilik teorisinin geliştiricisi olarak kavrar ve bu bağlamda yapılandırılmış bilgi sunar.

Google Bilgi Grafı'nın temel amacı, "anahtar kelime araması" ile "anlam tabanlı arama" arasındaki farkı ortadan kaldırmaktır. Geleneksel arama motorları, bir sorgudaki kelimeleri web sayfalarında arar. Ancak bu yöntem, aynı kavramın farklı ifadelerle yazıldığı durumlarda yetersiz kalır. Örneğin "ABD başkanı kimdir?" ve "United States president" sorguları aynı anlama gelir, ancak kelime düzeyinde farklıdır. Bilgi grafı bu tür eşdeğerlikleri çözerek sorguları varlık düzeyinde temsil eder.

Google Bilgi Grafı, milyarlarca varlık ve bu varlıklar arasındaki ilişkilerden oluşur. Bu varlıklar arasında kişiler, yerler, kurumlar, filmler, kitaplar, bilimsel kavramlar ve daha birçok kategori bulunur. Örneğin "Albert Einstein" varlığı, "Princeton Üniversitesi", "Özel Görelilik Teorisi" ve "Nobel Fizik Ödülü" gibi diğer varlıklarla ilişkili olarak modellenir. Böylece tek bir isim, geniş bir bilgi ağı içinde anlam kazanır.

Bu sistemin temel veri kaynakları oldukça çeşitlidir. Wikipedia, Wikidata, Freebase (tarihsel olarak), açık veri setleri ve web'den çıkarılan yapısal bilgiler bu grafın oluşturulmasında kullanılır. Ayrıca Google, kullanıcı etkileşimlerinden elde edilen anonim verileri de sistemin doğruluğunu artırmak için kullanır.

Bilgi grafının en önemli katkılarından biri, arama sonuçlarının zenginleştirilmesidir. Örneğin bir kişi "Leonardo da Vinci" araması yaptığında, yalnızca web sayfaları değil, onun doğum tarihi, eserleri, bağlantılı kişiler ve tarihi bağlamı da doğrudan arama sonuçlarında gösterilir. Bu yapı "bilgi paneli" (knowledge panel) olarak bilinir ve bilgi grafının doğrudan bir çıktısıdır.

Google Bilgi Grafı aynı zamanda yapay zekâ sistemleri için de kritik bir altyapıdır. Özellikle doğal dil işleme ve soru-cevap sistemlerinde varlık tabanlı temsil büyük önem taşır. Bir modelin "neyin ne olduğunu" anlayabilmesi için dünya bilgisinin yapılandırılmış olması gerekir.

Sonuç olarak Google Bilgi Grafı, internet üzerindeki dağınık bilgiyi anlamlı bir varlık-ilişki ağına dönüştürerek arama motorlarını bir "bilgi keşif sistemi" haline getirmiştir. Bu yaklaşım, yalnızca bilgiye erişimi kolaylaştırmakla kalmamış, aynı zamanda bilginin temsil biçimini de kökten değiştirmiştir.

7.2 Arama Motorları ve Anlamsal Arama

Arama motorları, internet üzerindeki en büyük bilgi erişim sistemleridir. Ancak klasik arama motorları uzun yıllar boyunca kelime temelli indeksleme (keyword indexing) yaklaşımıyla çalışmıştır. Bu yaklaşımda bir sorgu, web sayfalarındaki kelimelerle eşleştirilir ve en uygun sonuçlar sıralanır. Fakat bu yöntem, dilin doğasındaki anlam zenginliğini tam olarak yakalayamaz. İşte bu noktada anlamsal arama (semantic search) kavramı devreye girer.

Anlamsal arama, yalnızca kelimeleri değil, kelimelerin anlamlarını ve bağlamlarını da dikkate alan bir arama yaklaşımıdır. Örneğin kullanıcı "kalp hastalıkları tedavi yöntemleri" araması yaptığında, sistem yalnızca bu ifadeyi içeren sayfaları değil, "kardiyoloji", "kalp ameliyatı", "stent uygulaması" gibi semantik olarak ilişkili içerikleri de sonuçlara dahil eder.

Bu yaklaşımın temelinde bilgi grafı ve vektör temsilleri (gömme) bulunur. Vektör tabanlı sistemler, metinleri sayısal uzayda temsil ederek benzer anlamları yakın konumlara yerleştirir. Böylece “doktor” ve “hekim” gibi farklı kelimeler aynı anlamı temsil ettiği için benzer sonuçlar üretir.

Anlamsal arama sistemleri genellikle üç katmandan oluşur. İlk katman, kullanıcı sorgusunu analiz eder ve varlıkları tespit eder. İkinci katman, bu varlıkların bilgi grafı içindeki ilişkilerini kullanarak bağlam oluşturur. Üçüncü katman ise sonuçları sıralar ve kullanıcıya en anlamlı bilgiyi sunar.

Örneğin bir kullanıcı “Tesla elektrikli araçları” araması yaptığında sistem yalnızca Tesla’nın web sitesini değil, aynı zamanda Tesla’nın modelleri, rakip markalar, batarya teknolojileri ve ilgili haberleri de dikkate alır. Böylece kullanıcı yalnızca belge değil, anlamlı bir bilgi ağına erişir.

Anlamsal aramanın en önemli avantajlarından biri belirsizliği azaltmasıdır. Geleneksel sistemlerde aynı kelimenin farklı anlamları sorun yaratabilir. Örneğin “Java” kelimesi bir programlama dili, bir kahve türü veya bir ada olabilir. Anlamsal arama, bağlam analizi sayesinde doğru anlamı seçebilir.

Ayrıca bu sistemler kullanıcı niyetini anlamada da oldukça etkilidir. Kullanıcı bir bilgi mi arıyor, bir ürün mü satın almak istiyor, yoksa bir problem mi çözmeye çalışıyor, sistem bunu analiz edebilir.

Sonuç olarak arama motorları ve anlamsal arama teknolojileri, internet üzerindeki bilgiye erişim biçimini kökten değiştirmiştir. Kelime bazlı aramadan anlam bazlı aramaya geçiş, bilgi grafı ve yapay zekâ teknolojilerinin birleşimi sayesinde mümkün olmuştur. Bu dönüşüm, arama motorlarını yalnızca bilgi listeleyen sistemlerden, bilgiyi anlayan ve organize eden akıllı sistemlere dönüştürmüştür.

7.3 Öneri Sistemleri

Öneri sistemleri (recommendation systems), modern dijital ekonominin görünmez ama en güçlü motorlarından biridir. Netflix’te izlediğiniz dizilerden, Amazon’da karşınıza çıkan ürünlere, Spotify’da dinlediğiniz şarkılardan YouTube ana sayfanıza kadar pek çok deneyim bu sistemler tarafından şekillendirilir. Bilgi grafı yaklaşımı, öneri sistemlerini yalnızca “benzer kullanıcılar benzer şeyleri sever” mantığından çıkarıp, “varlıklar arasındaki ilişkileri anlayan sistemler” haline getirir.

Geleneksel öneri sistemleri genellikle iki ana yaklaşıma dayanır: içerik tabanlı filtreleme ve işbirlikçi filtreleme. İçerik tabanlı sistemler, bir kullanıcının daha önce beğendiği içeriklere benzer içerikleri önerir. İşbirlikçi filtreleme ise benzer kullanıcı davranışlarını analiz eder.

Örneğin “bu ürünü alanlar bunu da aldı” mantığıyla çalışır. Ancak bu yöntemler, özellikle soğuk başlangıç problemi (cold start problem)[12] ve bağlam eksikliği nedeniyle sınırlıdır.

Bilgi grafı tabanlı öneri sistemleri ise bu sınırlamaları aşmak için varlıklar arasındaki ilişkileri doğrudan modellemeye odaklanır. Örneğin bir film öneri sisteminde “oyuncu”, “yönetmen”, “tür”, “ödül” ve “izleyici yorumları” gibi farklı varlıklar bir graf içinde temsil edilir. “Christopher Nolan → yönetti → Inception” veya “Inception → türü → bilim kurgu” gibi ilişkiler, öneri mekanizmasının temelini oluşturur.

Bu yapı sayesinde sistem yalnızca kullanıcı davranışlarına değil, aynı zamanda içeriklerin anlam yapısına da bakabilir. Örneğin bir kullanıcı “Interstellar” filmini beğendiyse, sistem sadece diğer kullanıcıların tercihlerini değil, “uzay”, “zaman”, “Christopher Nolan” ve “Matthew McConaughey” gibi ilişkisel bağlantıları da kullanarak öneriler üretir.

Bilgi grafı tabanlı öneri sistemlerinin en önemli avantajlarından biri açıklanabilirliktir. Klasik derin öğrenme modelleri genellikle “kara kutu” olarak çalışırken, graf tabanlı sistemler bir önerinin neden yapıldığını açıkça gösterebilir. Örneğin sistem “size şu film önerildi çünkü aynı yönetmen ve benzer tema içeriyor” diyebilir. Bu durum kullanıcı güvenini önemli ölçüde artırır.

Bir diğer önemli avantaj bağlam zenginliğidir. Kullanıcı sadece geçmiş davranışlarına göre değil, ilgi alanları, sosyal bağlantılar ve içerik ilişkileri üzerinden de modellenebilir. Örneğin bir akademik öneri sisteminde “yazarlar”, “makaleler”, “atıf ilişkileri” ve “konu alanları” birlikte kullanılarak çok daha isabetli öneriler üretilebilir.

Modern sistemlerde bilgi grafı, vektör tabanlı modellerle birlikte kullanılır. Vektör temsilleri (gömme) kullanıcı ve içerik benzerliğini yakalarken, graf yapısı bu benzerlikleri ilişki ağı içinde anlamlandırır. Bu hibrit yaklaşım, hem ölçeklenebilir hem de anlamsal olarak güçlü öneri sistemleri oluşturur.

Örneğin bir e-ticaret platformunda kullanıcı “kablosuz kulaklık” aradığında, sistem yalnızca benzer ürünleri değil, aynı zamanda “marka ilişkileri”, “fiyat segmentleri” ve “kullanım senaryoları” gibi graf ilişkilerini de dikkate alarak öneri yapabilir. Bu sayede öneriler daha kişiselleştirilmiş ve bağlama uygun hale gelir.

Sonuç olarak bilgi grafı tabanlı öneri sistemleri, dijital deneyimi pasif bir içerik tüketiminden aktif bir anlam keşfine dönüştürür. Kullanıcı artık yalnızca geçmiş davranışlarına göre yönlendirilen bir sistemin parçası değil, ilişkisel bir bilgi ağının merkezinde yer alan aktif bir keşif aktörü haline gelir.

7.4 Finans ve Dolandırıcılık Analizi

Finans sektörü, veri yoğunluğu ve yüksek risk yapısı nedeniyle bilgi grafı uygulamaları için en kritik alanlardan biridir. Bankacılık işlemleri, ödeme sistemleri, sigorta kayıtları ve yatırım

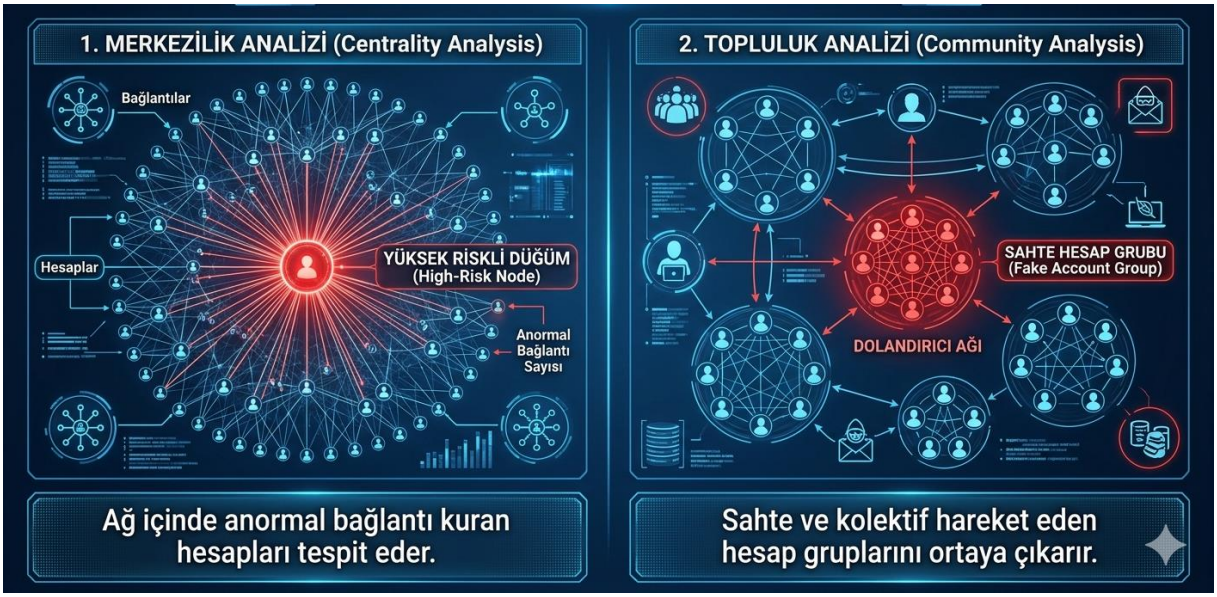
[12] Yapay zekâ tabanlı bir öneri sisteminin (örneğin Spotify'ın size şarkı önermesi veya Amazon'un ürün öneri etmesi), düzgün tahminler yapabilmesi için geçmiş verilere ihtiyacı vardır. Sistem bu verilerden yoksun olduğunda "soğuk başlangıç" yaşar.

hareketleri gibi veriler doğal olarak ilişkisel bir yapı taşır. Bu nedenle finansal sistemler, klasik tablo bazlı veri modellerinden ziyade graf tabanlı modellere oldukça uygundur.

Dolandırıcılık analizi (fraud detection), bilgi grafının en güçlü kullanım alanlarından biridir. Finansal dolandırıcılık genellikle tek bir işlem üzerinden değil, birden fazla işlem ve aktör arasındaki ilişkiler ağı üzerinden gerçekleşir. Örneğin aynı IP adresinden açılan birden fazla hesap, aynı banka kartının farklı ülkelerde kısa sürede kullanılması veya birbirine bağlı sahte şirket ağları, klasik analiz yöntemleriyle tespit edilmesi zor yapılardır.

Bilgi grafı bu noktada kritik bir avantaj sağlar çünkü tüm bu varlıkları ve ilişkileri tek bir ağ içinde temsil eder. Örneğin "Kullanıcı A → transfer etti → Kullanıcı B", "Kullanıcı B → sahip → Sahte Şirket X" gibi ilişkiler bir araya getirildiğinde gizli dolandırıcılık ağları ortaya çıkarılabilir. Bu tür ağ yapıları, klasik satır-sütun analizlerinde kolayca görünmez.

Şekil 21. Dolandırıcılık Tespiti: Ağ Analizi Yöntemleri



Şekil 21, sosyal ağ veya finansal işlem analitiğinde sıklıkla kullanılan iki temel graf algoritması yaklaşımını siber güvenlik ve dolandırıcılık tespiti (fraud detection) bağlamında ele almaktadır. Sol paneldeki Merkezilik Analizi (Centrality Analysis), ağ üzerindeki tekil aktörlere odaklanarak çok geniş bir kitleyle alışılmadık derecede yoğun bağlar kuran, ağın odağındaki "Yüksek Riskli Düğümü (High-Risk Node)" ve onun anormal bağlantı sayılarını cımbızla çeker gibi tespit eder. Sağ paneldeki Topluluk Analizi (Community Analysis) ise tekil hesaplar yerine yapısal davranışlara odaklanır; kendi aralarında son derece sıkı, kapalı ve organize ilişkiler kurarak kolektif bir biçimde hareket eden şüpheli kümeleri, yani bir "Dolandırıcı Ağı" içindeki "Sahte Hesap Grubunu (Fake Account Group)" ortaya çıkarır. Özetle şema, bütüncül bir graf yapısı üzerinde hem bireysel anomaliyi (merkezilik) hem de organize suç şebekelerini (topluluk) tespit etmenin metodolojisini görselleştirmektedir. Finansal sistemlerde graf analizinin en önemli gücü "çok adımlı ilişkileri" takip edebilmesidir. Bir işlem doğrudan şüpheli görünmeyebilir, ancak birkaç adım sonrası yüksek riskli bir yapıya bağlanabilir. Örneğin bir kullanıcıdan çıkan

para önce bir aracı hesaba, oradan offshore bir şirkete ve ardından farklı hesaplara dağılıyorsa, bu zincir graf analizi ile tespit edilebilir.

Merkezilik analizleri dolandırıcılık tespitinde sıkça kullanılır. Örneğin bir hesap çok sayıda farklı hesapla anormal şekilde bağlantı kuruyorsa, bu hesap ağ içinde "yüksek riskli düğüm" olarak işaretlenebilir. Benzer şekilde topluluk analizi yöntemleri, sahte hesap gruplarını ortaya çıkarabilir.

Bilgi grafı ayrıca finansal kurumlarda müşteri ilişkilerini anlamak için de kullanılır. Örneğin bir bankada bir müşterinin kredi geçmişi, işlem davranışları, sosyal bağlantıları ve şirket ilişkileri birlikte analiz edilerek daha doğru risk profili çıkarılabilir. Bu, kredi skorlama sistemlerinin daha isabetli hale gelmesini sağlar.

Modern dolandırıcılık sistemleri genellikle gerçek zamanlı çalışır. Bir işlem gerçekleştiği anda sistem graf üzerinde bu işlemin bağlantılarını analiz eder ve risk skorunu hesaplar. Eğer belirli bir eşik aşılırsa işlem durdurulabilir veya ek doğrulama istenebilir. Bu tür sistemler, özellikle yüksek hacimli ödeme ağlarında kritik öneme sahiptir.

Ayrıca bilgi grafı, kara para aklama gibi karmaşık finansal suçların tespitinde de kullanılır. Bu tür suçlar genellikle çok katmanlı ve gizli transfer ağları içerir. Graf yapılar, bu gizli ağları görünür hale getirerek analiz edilmesini sağlar.

Sonuç olarak finans ve dolandırıcılık analizi, bilgi grafı teknolojisinin en güçlü uygulama alanlarından biridir. Graf tabanlı yaklaşım, finansal veriyi yalnızca sayısal bir kayıt sistemi olmaktan çıkarır ve onu ilişkisel bir risk ağına dönüştürür. Bu sayede hem güvenlik artar hem de finansal karar süreçleri daha akıllı ve açıklanabilir hale gelir.

7.5 Sağlık ve İlaç Keşfi

Sağlık sektörü, veri yoğunluğu ve yüksek doğruluk ihtiyacı nedeniyle bilgi grafı uygulamalarının en kritik ve en yüksek etki alanlarından biridir. Hastalıklar, semptomlar, ilaçlar, genetik bilgiler, klinik denemeler ve hasta kayıtları doğal olarak birbirine bağlı yapılar içerir. Bu nedenle sağlık verisini düz tablolar yerine ilişkisel bir ağ olarak modellemek, hem tanı süreçlerini hem de ilaç geliştirme çalışmalarını ciddi biçimde dönüştürür.

Bilgi grafı yaklaşımında bir hastalık yalnızca bir etiket değil, çok boyutlu bir varlık olarak ele alınır. Örneğin "Diyabet" düğümü; "insülin direnci", "pankreas fonksiyonu", "obezite", "kardiyovasküler hastalıklar" ve "ilaç tedavileri" gibi birçok düğümle ilişkilidir. Bu ilişkiler sayesinde sistem yalnızca hastalığı tanımlamakla kalmaz, aynı zamanda hastalığın biyolojik ve klinik bağlamını da temsil eder.

İlaç keşfi süreci bilgi grafı ile birlikte çok daha sistematik hale gelir. Geleneksel yöntemlerde ilaç geliştirme uzun, maliyetli ve düşük başarı oranına sahip bir süreçtir. Ancak bilgi grafı, biyolojik veriler arasındaki gizli ilişkileri ortaya çıkararak bu süreci hızlandırabilir. Örneğin bir ilaç molekülü ile belirli bir protein arasındaki etkileşim, graf içinde açık bir ilişki olarak temsil edilir. Bu sayede potansiyel yeni ilaç adayları daha hızlı tespit edilebilir.

Örnek olarak, “Aspirin → inhibe eder → COX-1 enzimi” ilişkisi, ağ içinde diğer enzimler ve hastalıklarla bağlantılı olarak analiz edilebilir. Bu analiz sonucunda Aspirin’in yalnızca ağrı kesici değil, aynı zamanda kardiyovasküler risk azaltıcı etkisi de graf olarak görülebilir.

Bilgi grafı ayrıca hastalık teşhisinde de önemli bir rol oynar. Bir hastanın semptomları graf üzerinde ilgili hastalık düğümleriyle eşleştirilir. Örneğin “ateş”, “öksürük” ve “nefes darlığı” semptomları COVID-19, grip veya zatürre gibi hastalıklarla bağlantılı olabilir. Sistem bu ilişkileri analiz ederek olası teşhisleri sıralayabilir.

Genetik verilerle bilgi grafının birleşimi, biyoinformatik alanında büyük bir devrim yaratmıştır. Genler, proteinler ve hastalıklar arasındaki ilişkiler graf yapılarla modellenerek karmaşık biyolojik sistemler anlaşılabilir hale gelir. Örneğin “BRCA1 geni → ilişkilidir → meme kanseri” gibi bağlantılar, genetik risk analizlerinde kritik rol oynar.

Klinik araştırmalar da bilgi grafi ile daha verimli hale gelir. Farklı hastalıklar ve tedavi yöntemleri arasındaki ilişkiler analiz edilerek yeni tedavi protokolleri geliştirilebilir. Ayrıca hasta kayıtları graf içinde anonimleştirilmiş şekilde analiz edilerek büyük ölçekli sağlık trendleri ortaya çıkarılabilir.

Modern sağlık sistemlerinde bilgi grafı, yapay zekâ ile birlikte kullanılarak karar destek sistemleri oluşturur. Doktorlar, hastanın verilerini sisteme girdiklerinde graf tabanlı analiz sayesinde olası teşhisler ve tedavi seçenekleri sunulur. Bu, özellikle karmaşık vakalarda ikinci bir uzman görüşü gibi çalışır.

Sonuç olarak sağlık ve ilaç keşfi alanında bilgi grafı, yalnızca veri yönetimi aracı değil, aynı zamanda yaşam kurtarıcı bir karar destek mekanizmasıdır. Hastalıklar, ilaçlar ve biyolojik süreçler arasındaki karmaşık ilişkileri görünür hale getirerek hem bilimsel keşifleri hızlandırır hem de klinik kararların doğruluğunu artırır.

7.6 Endüstri 4.0 ve Dijital İkizler

Endüstri 4.0, üretim sistemlerinin dijitalleşmesi ve akıllı hale gelmesi sürecini ifade eder. Bu dönüşümün merkezinde sensörler, nesnelerin interneti (IoT), büyük veri ve yapay zekâ yer alır. Ancak bu teknolojilerin gerçek potansiyelini ortaya çıkaran yapı, bilgi grafi yaklaşımıdır. Çünkü endüstriyel sistemler doğası gereği karmaşık, ilişkisel ve dinamik yapılardır.

Dijital ikiz (digital twin) kavramı, fiziksel bir sistemin dijital ortamda birebir temsil edilmesini ifade eder. Bir fabrikanın makineleri, üretim hatları, enerji tüketimi ve bakım süreçleri dijital ortamda sürekli güncellenen bir modelle temsil edilir. Bu model, bilgi grafi ile birleştiğinde çok daha güçlü hale gelir. Çünkü artık yalnızca nesneler değil, bu nesneler arasındaki ilişkiler de modellenir.

Örneğin bir üretim hattında “Makine A → üretir → Ürün X” ve “Makine A → bağlıdır → Sensör B” gibi ilişkiler graf içinde temsil edilir. Sensör B’den gelen veri, Makine A’nın performansını doğrudan etkiler. Bu ilişkiler sayesinde sistem, arızaları önceden tahmin edebilir ve bakım süreçlerini optimize edebilir.

Bilgi grafı tabanlı dijital ikizler, gerçek zamanlı veri akışı ile sürekli güncellenir. Sensörlerden gelen sıcaklık, basınç, hız ve enerji tüketimi gibi veriler graf yapısına entegre edilir. Böylece sistem yalnızca geçmişi analiz etmekle kalmaz, aynı zamanda gelecekteki olası durumları da tahmin edebilir.

Endüstriyel bakım (predictive maintenance) bu yaklaşımın en önemli kullanım alanlarından biridir. Örneğin bir motorun titreşim verileri normalden sapmaya başladığında, sistem bu durumu graf üzerinden analiz ederek olası arızayı önceden tespit edebilir. Bu sayede üretim duruşları minimize edilir ve maliyetler düşer.

Tedarik zinciri yönetimi de bilgi grafı ile büyük ölçüde iyileştirilir. Ürünler, tedarikçiler, lojistik ağları ve depolar arasındaki ilişkiler graf içinde modellenerek tüm sistemin görünürlüğü sağlanır. Örneğin bir tedarikçide yaşanan gecikme, tüm üretim hattını nasıl etkileyeceği graf üzerinden analiz edilebilir.

Enerji yönetimi de dijital ikizlerin önemli bir uygulama alanıdır. Fabrika içindeki enerji tüketimi, makineler arasındaki ilişkilere göre optimize edilebilir. Örneğin yüksek enerji tüketen bir makine, düşük talep dönemlerinde devre dışı bırakılabilir veya farklı bir üretim planına dahil edilebilir.

Bilgi grafı ayrıca endüstriyel güvenlik sistemlerinde de kullanılır. Sensör verileri ve makine durumları analiz edilerek olası riskler önceden tespit edilir. Örneğin bir gaz sızıntısı sensörü ile bir üretim hattı arasındaki ilişki, acil durum senaryolarını otomatik olarak tetikleyebilir.

Sonuç olarak Endüstri 4.0 ve dijital ikizler, bilgi grafı teknolojisinin en güçlü endüstriyel uygulamalarından biridir. Fiziksel dünyayı dijital bir ağ yapısına dönüştürerek üretim sistemlerini daha akıllı, daha esnek ve daha öngörülebilir hale getirir. Bu yaklaşım, geleceğin fabrikalarını yalnızca otomatik değil, aynı zamanda “anlayan sistemler” haline getirmektedir.

7.7 Savunma ve İstihbarat Sistemleri

Savunma ve istihbarat sistemleri, bilgi grafı yaklaşımının en kritik, en karmaşık ve aynı zamanda en yüksek güvenlik gerektiren kullanım alanlarından biridir. Bu alanlarda veri yalnızca çok büyük hacimde değildir; aynı zamanda parçalı, heterojen, zaman bağımlı ve çoğu zaman belirsizlik içeren bir yapıdadır. Bu nedenle klasik veri tabanı yaklaşımları, özellikle ilişkisel (tablosal) modeller, bu karmaşık yapıyı temsil etmekte yetersiz kalır. Bilgi grafı ise bu tür ortamlarda varlıklar ve ilişkiler üzerinden bütüncül bir durum resmi oluşturma imkânı sağlar.

Savunma sistemlerinde temel problem “durum farkındalığı”dır (situational awareness). Bir operasyon sahasında insanlar, araçlar, iletişim sinyalleri, coğrafi bölgeler ve zaman olayları sürekli değişir. Örneğin bir bölgede “Araç A → hareket ediyor → Bölge X” bilgisi, “Araç A → temas kurdu → Araç B” ilişkisiyle birleştiğinde potansiyel bir operasyonel durum ortaya çıkar. Bu tür ilişkiler ancak graf yapılarla anlamlı hale gelir.

İstihbarat sistemlerinde bilgi grafı, farklı kaynaklardan gelen verileri birleştirmek için kullanılır. Uydu görüntüleri, sinyal istihbaratı (SIGINT), insan istihbaratı (HUMINT) ve açık kaynak

istihbaratı (OSINT) gibi farklı veri türleri tek bir graf içinde birleştirilebilir. Örneğin bir telefon sinyali, belirli bir kişiyle ilişkilendirildiğinde ve bu kişi belirli bir araçla aynı bölgede tespit edildiğinde, sistem bu ilişkileri graf olarak bağlayabilir.

Bir örnek üzerinden düşünelim: “Kişi A → iletişim kurdu → Kişi B”, “Kişi B → ziyaret etti → Lokasyon Y”, “Lokasyon Y → bağlantılı → Şüpheli Organizasyon Z”. Bu üçlü yapı, klasik veri analizinde parçalı bilgiler olarak kalırken, graf sistemde birleşik bir tehdit modeli oluşturur.

Bilgi grafı ayrıca tehdit analizi ve anomali tespitinde de güçlü bir araçtır. Normal davranış kalıpları graf üzerinde modellenir ve bu kalıplardan sapmalar otomatik olarak tespit edilir. Örneğin bir askeri araç normalde belirli rotalarda hareket ediyorsa, bu rotanın dışına çıkması sistem tarafından anomali olarak işaretlenebilir.

Zaman boyutu savunma sistemlerinde kritik bir rol oynar. Olaylar sadece “ne oldu” sorusuyla değil, “ne zaman oldu” sorusuyla birlikte değerlendirilir. Bu nedenle zaman damgalı graflar, istihbarat analizlerinin temel bileşenidir. Örneğin bir iletişim sinyali, bir operasyon öncesi veya sonrası gerçekleştiğine göre tamamen farklı anlamlar taşıyabilir.

Bilgi grafı ayrıca komuta-kontrol sistemlerinde de kullanılır. Farklı birimler arasındaki iletişim, kaynaklar, görevler ve operasyonlar graf yapılarla modellenerek daha hızlı karar alma süreçleri oluşturulur. Bu sayede karmaşık operasyonlar daha koordine bir şekilde yönetilebilir.

Modern savunma sistemlerinde yapay zekâ ile bilgi grafı birlikte çalışır. Graf tabanlı modeller, yapay zekâya bağlam sağlar. Örneğin bir model yalnızca “hareket eden araçlar” değil, bu araçların hangi birliklere ait olduğunu, hangi görevde olduklarını ve hangi risk seviyesinde bulunduklarını da bilir.

Sonuç olarak savunma ve istihbarat sistemlerinde bilgi grafı, yalnızca bir veri yapısı değil, aynı zamanda operasyonel zekânın temel altyapısıdır. Dağınık veriyi anlamlı ilişkilere dönüştürerek karmaşık tehdit ortamlarını görünür hale getirir ve karar alma süreçlerini hızlandırır.

7.8 Akademik Bilgi Ağları

Akademik dünya, doğal olarak ilişkisel bir bilgi yapısına sahiptir. Makaleler, yazarlar, atıflar, konferanslar, dergiler ve araştırma alanları birbirine sıkı şekilde bağlıdır. Bu nedenle akademik sistemler, bilgi grafı yaklaşımı için en uygun veri ekosistemlerinden birini oluşturur. Akademik bilgi ağları, bilimin nasıl üretildiğini, nasıl yayıldığını ve nasıl geliştiğini anlamak için güçlü bir araçtır.

Akademik bilgi ağlarının en temel bileşeni atıf ilişkileridir. Bir makale başka bir makaleyi referans gösterdiğinde, bu iki makale arasında yönlü bir ilişki oluşur. Örneğin “Makale A → atıf yapar → Makale B” ilişkisi, bilimsel bilginin zaman içindeki akışını temsil eder. Bu yapı, bilimsel bilginin bir ağ olarak evrimini anlamamıza olanak sağlar.

Yazarlar da bu ağın önemli düğümleridir. Bir yazarın hangi makaleleri yazdığı, hangi araştırma alanlarında çalıştığı ve hangi diğer yazarlarla iş birliği yaptığı graf yapılarla modellenebilir. Örneğin "Yazar X → birlikte yazdı → Yazar Y" ilişkisi, bilimsel iş birliği ağlarını ortaya çıkarır.

Akademik bilgi grafı aynı zamanda konu (yapısını da içerir. Örneğin "makine öğrenmesi", "doğal dil işleme" ve "bilgisayarlı görü" gibi alanlar birbirleriyle ilişkili alt alanlar olarak graf içinde temsil edilir. Bu sayede bir araştırma alanının diğer alanlarla nasıl bağlantılı olduğu görülebilir.

Bir örnek üzerinden düşünelim: "Makale A → konu → Graf Neural Networks", "Makale A → atıf yapar → Makale B", "Makale B → yazar → Araştırmacı Z". Bu yapı sayesinde yalnızca makaleler değil, aynı zamanda fikirlerin ve araştırmacıların ilişkileri de görünür hale gelir.

Akademik bilgi ağlarının en önemli kullanım alanlarından biri literatür keşfidir. Araştırmacılar belirli bir konuda yeni makaleler ararken, bilgi grafı tabanlı sistemler en ilgili çalışmaları sadece anahtar kelimelere göre değil, ilişkisel bağlamlara göre önerir. Örneğin bir araştırmacı "transformer modelleri" ile ilgileniyorsa, sistem yalnızca bu kelimeyi içeren makaleleri değil, bu modellerle ilişkili tüm çalışmalarını da sunabilir.

Atıf ağları aynı zamanda bilimsel etki analizi için de kullanılır. Bir makalenin veya yazarın bilimsel etkisi, sadece yayın sayısıyla değil, ağ içindeki konumuyla da ölçülebilir. Örneğin yüksek derecede bağlantılı ve sık atıf alan düğümler, bilimsel açıdan daha etkili olarak değerlendirilebilir.

Bilgi grafı ayrıca bilim haritaları oluşturmak için de kullanılır. Bu haritalar, bilimsel alanların nasıl geliştiğini ve hangi alanların birbirine yakın olduğunu görselleştirir. Örneğin yapay zekâ ile biyoinformatik arasındaki bağlantılar, graf üzerinde açıkça görülebilir.

Modern akademik sistemlerde bu yapı, yapay zekâ destekli araştırma asistanlarıyla birlikte kullanılır. Bu sistemler araştırmacıya yalnızca makale önermekle kalmaz, aynı zamanda araştırma boşluklarını da tespit edebilir.

Sonuç olarak akademik bilgi ağları, bilimi yalnızca bir yayın koleksiyonu olarak değil, dinamik ve yaşayan bir bilgi ağı olarak görmemizi sağlar. Bu yaklaşım, bilimsel keşif sürecini hızlandırır ve bilginin evrimini daha anlaşılır hale getirir.

7.9 Akıllı Şehirler

Akıllı şehirler, şehir yaşamını daha verimli, sürdürülebilir ve yaşanabilir hale getirmek için veri, sensörler ve yapay zekâ teknolojilerinin birlikte kullanıldığı sistemlerdir. Bu sistemlerin merkezinde giderek daha fazla bilgi grafı yaklaşımı yer almaktadır. Çünkü şehir dediğimiz yapı, doğası gereği bir "ilişkiler ağıdır": insanlar, ulaşım sistemleri, enerji ağları, binalar, altyapı ve hizmetler sürekli birbirleriyle etkileşim halindedir.

Geleneksel şehir yönetimi, çoğunlukla birbirinden bağımsız veri silolarına dayanır. Ulaşım verisi ayrı, enerji tüketimi ayrı, güvenlik kayıtları ayrı sistemlerde tutulur. Bu durum, şehirde olup bitenleri bütüncül olarak anlamayı zorlaştırır. Bilgi grafı yaklaşımı ise bu farklı veri kaynaklarını

tek bir ilişkisel model içinde birleştirir. Örneğin “Otobüs Hattı 5 → hizmet verir → Bölge A” ve “Bölge A → yüksek trafik yoğunluğu → 08:00–09:00” gibi ilişkiler aynı graf içinde temsil edilebilir.

Akıllı şehirlerde bilgi grafı özellikle ulaşım optimizasyonunda önemli bir rol oynar. Trafik sensörleri, GPS verileri ve toplu taşıma sistemleri birlikte analiz edilerek şehir içi hareketlilik modellenir. Örneğin bir bölgede trafik yoğunluğu arttığında sistem, alternatif güzergâhları graf üzerinden hesaplayabilir ve sürücülere veya toplu taşıma sistemlerine öneriler sunabilir. Bu yaklaşım, sadece anlık çözüm üretmekle kalmaz, aynı zamanda uzun vadeli planlamayı da destekler.

Enerji yönetimi de akıllı şehirlerin kritik bir bileşenidir. Elektrik şebekeleri (smart grids), üretim ve tüketim noktaları arasındaki ilişkiler üzerinden modellenebilir. Örneğin “Bina X → tüketir → elektrik”, “Güneş Paneli Y → üretir → elektrik” ilişkileri graf üzerinde analiz edilerek enerji dengesi optimize edilebilir. Talep yoğunluğuna göre enerji dağıtımı dinamik olarak ayarlanabilir.

Güvenlik ve acil durum yönetimi de bilgi grafı ile güçlendirilir. Şehirdeki kameralar, sensörler ve olay kayıtları birbirine bağlanarak gerçek zamanlı bir durum farkındalığı oluşturulur. Örneğin bir bölgede “yangın alarmı → aktif”, “duman sensörü → tetiklenmiş” ve “itfaiye aracı → yolda” gibi bilgiler birleşerek bütüncül bir kriz yönetimi sağlanabilir.

Akıllı şehirlerde vatandaş davranışları da graf yapılarla analiz edilebilir. İnsanların hareket kalıpları, sosyal etkileşimleri ve hizmet kullanım alışkanlıkları anonimleştirilmiş şekilde modele dahil edilebilir. Bu sayede şehir planlaması daha veri odaklı hale gelir. Örneğin bir bölgede sürekli yoğunlaşan insan hareketi varsa, yeni bir metro hattı planlanabilir.

Bilgi grafı aynı zamanda farklı şehir sistemleri arasında entegrasyon sağlar. Ulaşım, sağlık, eğitim ve belediye hizmetleri tek bir ekosistem gibi çalışabilir. Örneğin bir vatandaşın sağlık randevusu ile ulaşım planlaması entegre edilerek daha verimli bir şehir deneyimi sunulabilir.

Sonuç olarak akıllı şehirler, yalnızca sensörlerle donatılmış dijital altyapılar değildir; aynı zamanda ilişkisel düşünceye dayalı büyük bilgi sistemleridir. Bilgi grafı bu sistemlerin “beyni” gibi çalışarak şehirleri daha uyumlu, verimli ve sürdürülebilir hale getirir.

7.10 Sosyal Ağ Analitiği

Sosyal ağ analitiği, bireyler, gruplar ve kurumlar arasındaki ilişkilerin incelenmesini sağlayan güçlü bir yöntemdir. Bu alanın temelinde de bilgi grafı yaklaşımı yer alır. Çünkü sosyal yapılar doğası gereği düğümler (insanlar) ve bağlantılar (ilişkiler) üzerinden modellenebilir.

Sosyal ağlar, yalnızca kim kiminle bağlantılı sorusunu değil, aynı zamanda bu bağlantıların gücünü, yönünü ve anlamını da içerir. Örneğin “Ahmet → arkadaş → Mehmet” ilişkisi basit bir bağlantıyı ifade ederken, “Ahmet → iş birliği yaptı → Mehmet” ilişkisi daha profesyonel bir bağlamı temsil eder. Bu tür farklı ilişki türleri graf yapılarla açık şekilde modellenebilir.

Sosyal ağ analitiğinin en temel kavramlarından biri merkezilik ölçüleridir. Bir ağda hangi düğümlerin daha önemli olduğunu belirlemek için derece merkeziliği, yakınlık merkeziliği ve aradalık merkeziliği gibi ölçüler kullanılır. Örneğin çok sayıda kişiyle bağlantısı olan bir kullanıcı, ağ içinde daha etkili bir konuma sahip olabilir.

Bir diğer önemli kavram topluluk yapılarıdır. Sosyal ağlarda insanlar genellikle kümeler halinde organize olur. aileler, iş arkadaşları, okul grupları veya ilgi toplulukları bu kümelere örnektir. Bilgi grafı bu kümeleri otomatik olarak tespit ederek sosyal yapının alt organizasyonlarını ortaya çıkarabilir.

Sosyal ağ analitiği aynı zamanda bilgi yayılımı süreçlerini anlamak için de kullanılır. Bir bilginin ağ içinde nasıl yayıldığı, hangi düğümler üzerinden hızlandığı ve nerede durduğu analiz edilebilir. Örneğin bir haberin sosyal medyada viral hale gelmesi, belirli "etkili düğümler" üzerinden gerçekleşir.

Örnek olarak bir Twitter (X) paylaşımını düşünelim: Bir kullanıcı bir mesaj paylaşır, bu mesaj bazı kullanıcılar tarafından yeniden paylaşılır ve bu süreç zincirleme şekilde büyür. Graf analiz bu yayılımın hangi kullanıcılar tarafından hızlandırıldığını ortaya çıkarabilir.

Sosyal ağ analitiği aynı zamanda sahte hesap (bot) tespiti için de kullanılır. Anormal bağlantı desenleri, olağandışı etkileşim yoğunluğu ve tekrar eden davranış kalıpları graf üzerinde kolayca tespit edilebilir. Örneğin birbirine aşırı yoğun şekilde bağlı ama dış dünya ile az bağlantısı olan bir grup, bot ağı olabilir.

İş dünyasında sosyal ağ analitiği, organizasyon içi iletişimi anlamak için kullanılır. Bir şirket içinde kimlerin daha merkezi rol oynadığı, bilgi akışının nasıl gerçekleştiği ve hangi ekiplerin daha izole olduğu graf analizle ortaya çıkarılabilir.

Sonuç olarak sosyal ağ analitiği, insan davranışlarını yalnızca bireysel düzeyde değil, ilişkisel bir sistem olarak anlamamızı sağlar. Bilgi grafı bu yapıyı matematiksel ve hesaplanabilir hale getirerek sosyal dünyanın görünmeyen dinamiklerini açığa çıkarır.

BÖLÜM 8 — BİLGİ GRAFI VE BİLİM

Bilim, özünde bir bilgi üretim ve örgütleme sürecidir. Bu süreç yalnızca yeni bulguların elde edilmesinden ibaret değildir; aynı zamanda bu bulguların önceki bilgilerle nasıl ilişkilendiğini anlamayı da içerir. Bu nedenle bilimsel bilgi, doğası gereği ilişkisel bir yapıya sahiptir. Bilgi grafı yaklaşımı, bilimi bu ilişkisel doğasına uygun biçimde modellemek için güçlü bir çerçeve sunar.

Geleneksel bilimsel yayın sistemi, makaleleri çoğunlukla bağımsız belgeler olarak ele alır. Ancak gerçekte her makale, diğer makalelerle, yazarlarla, kavramlarla ve deneysel verilerle bağlantılıdır. Örneğin bir fizik makalesi, Newton mekaniğinden kuantum teorisine kadar uzanan bir bilgi ağı içinde konumlanır. Bu bağlantılar açıkça görünmediğinde bilimsel bilgi parçalı kalır. Bilgi grafı, bu parçaları bir araya getirerek bilimsel bilginin bütünsel bir haritasını oluşturur.

Bu bölümde bilimsel bilginin nasıl graf yapılarla temsil edilebileceği ele alınacaktır. Atıf ağları, araştırmacı ağları ve konu ağları gibi yapılar üzerinden bilimin dinamik yapısı incelenecektir. Örneğin bir makalenin “atıf yaptığı çalışmalar”, “benzer konu alanları” ve “ortak yazarları” aynı graf içinde temsil edilebilir.

Ayrıca bilimsel keşif süreçlerinde bilgi grafının nasıl kullanılabileceği tartışılacaktır. Gizli bağlantıların ortaya çıkarılması, disiplinler arası ilişkilerin analiz edilmesi ve yeni araştırma alanlarının keşfi gibi konular bu çerçevede değerlendirilecektir. Bilgi grafı, bilimi yalnızca geçmişin bir kaydı olarak değil, geleceğin olasılıklarını da içeren dinamik bir sistem olarak görmemizi sağlar. Sonuç olarak bu bölüm, bilimin kendisini bir bilgi ağı olarak ele alarak, bilgi grafı yaklaşımının bilimsel düşünceyi nasıl dönüştürdüğünü göstermeyi amaçlamaktadır.

8.1 Bilimsel Keşiflerde Ağ Yaklaşımı

Bilimsel keşif, uzun süre boyunca bireysel deha, deneysel gözlem ve teorik modelleme ekseninde anlatıldı. Ancak modern bilim, giderek daha fazla "kolektif bir ağ sistemi" olarak anlaşılmaktadır. Bu perspektifte bilgi, izole birimler halinde değil; kavramlar, makaleler, araştırmacılar, kurumlar ve deneyler arasındaki ilişkiler ağı içinde ortaya çıkar. İşte bu noktada ağ yaklaşımı, bilimsel keşfi anlamak için güçlü bir çerçeve sunar.

Ağ yaklaşımı, bilimi bir düğümler ve bağlantılar sistemi olarak ele alır. Düğümler; araştırmacılar, makaleler, kavramlar, laboratuvarlar veya deneysel sonuçlar olabilir. Bağlantılar ise atıf ilişkileri, ortak yazarlık, kavramsal benzerlik veya deneysel bağımlılıklar şeklinde tanımlanabilir. Örneğin "Makale A → atıf yapar → Makale B" ilişkisi yalnızca iki yayın arasındaki bağlantıyı değil, aynı zamanda bilgi akışının yönünü de gösterir.

Bu yaklaşım, bilimsel keşfi doğrusal bir süreç olmaktan çıkarır. Geleneksel modelde bilim, "problem → hipotez → deney → sonuç" şeklinde ilerleyen doğrusal bir zincir olarak görülür. Oysa gerçek bilimsel süreçler çok daha karmaşıktır; geri beslemeler, paralel gelişmeler ve disiplinler arası etkileşimler içerir. Ağ modeli bu karmaşıklığı doğal biçimde temsil eder.

Örneğin yapay zekâ alanındaki gelişmeler, matematik, istatistik, bilgisayar bilimi ve nörobilim gibi farklı alanların kesişiminde ortaya çıkmıştır. Bu alanlar arasındaki bağlantılar bir ağ yapısında gösterildiğinde, bilimin nasıl disiplinler arası bir yapıya dönüştüğü açıkça görülebilir. "Derin öğrenme" kavramı bile tek bir alana ait değil, birçok alanın kesişim noktasında doğmuştur.

Ağ yaklaşımı ayrıca bilimsel yeniliğin nasıl yayıldığını anlamak için de kullanılır. Bir fikir ortaya çıktıktan sonra hangi araştırmacılar tarafından benimsendiği, hangi alanlara yayıldığı ve nasıl evrildiği graf yapılar üzerinden analiz edilebilir. Bu süreç "bilgi difüzyonu" olarak adlandırılır.

Bilimsel ağlarda merkezi düğümler genellikle büyük etkiye sahip fikirleri veya araştırmacıları temsil eder. Ancak sadece merkezilik değil, ağ içindeki köprü düğümler de kritik öneme sahiptir. Bu düğümler farklı alanlar arasında bilgi transferini sağlar. Örneğin fizik ile biyoloji arasında çalışan bir araştırmacı, iki alan arasında bilgi köprüsü oluşturabilir.

Ayrıca ağ yaklaşımı, bilimsel keşiflerde "görünmeyen bağlantıların" ortaya çıkarılmasını sağlar. Farklı disiplinlerde yapılan çalışmalar arasında doğrudan atıf olmasa bile, kavramsal benzerlikler graf analizle tespit edilebilir. Bu durum yeni araştırma alanlarının keşfine yol açabilir.

Sonuç olarak bilimsel keşiflerde ağ yaklaşımı, bilimi bireysel üretimden kolektif bir bilgi sistemi haline getirir. Bu perspektif, bilimi daha dinamik, ilişkisel ve evrimsel bir süreç olarak anlamamıza olanak tanır.

8.2 Atıf Ağları ve Bilim Haritaları

Atıf ağları, bilimsel bilginin zaman içindeki akışını ve evrimini anlamak için kullanılan en önemli graf yapılardan biridir. Bu ağlarda her düğüm bir bilimsel makaleyi temsil ederken, her bağlantı bir atıf ilişkisini gösterir. Yani "Makale A → atıf yapar → Makale B" ifadesi, bilginin B'den A'ya doğru aktığını gösterir. Bu yapı, bilimin tarihsel gelişimini izlemek için son derece güçlü bir araçtır.

Atıf ağlarının en önemli özelliklerinden biri yönlü yapıya sahip olmalarıdır. Çünkü bir atıf ilişkisi her zaman belirli bir yön taşır. Yeni bir makale eski bir makaleye atıf yapabilir, ancak tersi mümkün değildir. Bu yönlü yapı, bilginin zaman içindeki akışını modellemeyi mümkün kılar.

Atıf ağları üzerinden yapılan analizler, bilimsel etkinin ölçülmesinde önemli rol oynar. Bir makale ne kadar çok atıf alıyorsa, genellikle o kadar etkili kabul edilir. Ancak yalnızca atıf sayısı değil, atıfların geldiği düğümlerin konumu da önemlidir. Örneğin merkezi ve etkili düğümlerden gelen atıflar, daha yüksek bilimsel etki göstergesi olabilir.

Bilim haritaları, atıf ağlarının görselleştirilmiş ve analitik versiyonlarıdır. Bu haritalar, bilimsel alanların nasıl yapılandığını, hangi alanların birbirine yakın olduğunu ve hangi alanların hızla büyüdüğünü gösterir. Örneğin "makine öğrenmesi" ile "biyoinformatik" alanlarının kesiştiği bölgeler, yeni araştırma alanlarını temsil edebilir.

Bir örnek üzerinden düşünelim: Eğer "Makale A → atıf yapar → Makale B" ve "Makale B → atıf yapar → Makale C" ilişkileri varsa, bu zincir bilginin evrimsel bir yolunu gösterir. Bu zincirler bir araya geldiğinde bilimsel bilginin büyük ölçekli bir haritası ortaya çıkar.

Atıf ağları aynı zamanda "bilimsel patlama noktalarını" tespit etmek için de kullanılır. Bir makale kısa sürede çok sayıda atıf alıyorsa, bu genellikle alanı değiştiren bir keşif olduğunu gösterir. Örneğin derin öğrenme ile ilgili temel makaleler, kısa sürede büyük bir atıf yoğunluğu oluşturmuştur.

Bilim haritaları ayrıca araştırma alanlarının evrimini de gösterir. Bazı alanlar zamanla küçülürken, bazıları hızla büyür. Bu değişimler graf üzerinde açıkça görülebilir. Örneğin yapay zekâ alanı son yıllarda çok hızlı büyüyen bir alan olarak haritalarda belirgin hale gelmiştir.

Ayrıca atıf ağları, akademik iş birliği yapılarının analizinde de kullanılır. Hangi araştırmacıların birlikte çalıştığı, hangi kurumların daha merkezi olduğu ve hangi ülkelerin bilimsel üretimde daha etkili olduğu bu ağlar üzerinden analiz edilebilir.

Sonuç olarak atıf ağları ve bilim haritaları, bilimi statik bir bilgi birikimi olmaktan çıkarıp dinamik bir bilgi ekosistemi olarak görmemizi sağlar. Bu yaklaşım, bilimsel üretimin hem tarihsel hem de yapısal boyutlarını anlamada güçlü bir araç sunar.

8.3 Nobel Ödülleri ve Bilgi Ağları

Nobel Ödülleri, modern bilimin en prestijli göstergelerinden biri olarak kabul edilir. Ancak bu ödüller yalnızca bireysel başarıların bir listesi değildir; aynı zamanda bilimsel bilginin nasıl

örgütlendiğini ve hangi fikirlerin birbirini besleyerek büyük atılımlara dönüştüğünü gösteren dev bir bilgi ağının görünür yüzüdür. Bilgi grafı yaklaşımıyla bakıldığında Nobel Ödülleri, tekil olaylar değil; düğümler (kişiler, kurumlar, keşifler) ve bağlantılar (iş birlikleri, atıflar, etki ilişkileri) üzerinden birbirine bağlanan bir sistem olarak okunabilir.

Nobel Ödülü sahiplerini birer düğüm olarak düşündüğümüzde, bu düğümler arasındaki bağlantılar oldukça zengin bir ağ yapısı oluşturur. Örneğin aynı laboratuvarda çalışan araştırmacılar, birbirine atıf yapan makaleler veya aynı teorik çerçevede çalışan bilim insanları güçlü bağlarla birbirine bağlanır. Bu ağ, bilimin nasıl kolektif bir üretim olduğunu açıkça gösterir. Nadiren bir Nobel Ödülü tamamen izole bir keşfe dayanır; çoğu zaman uzun bir araştırma zincirinin sonucudur.

Örneğin 2013 Nobel Kimya Ödülü (Martin Karplus, Michael Levitt, Arieh Warshel) hesaplamalı kimyanın gelişimiyle doğrudan ilişkilidir. Bu üç bilim insanı arasındaki bağlantılar yalnızca ortak çalışma değil, aynı zamanda farklı alt disiplinlerin birleşimini de temsil eder. Bilgi grafı perspektifinden bakıldığında bu ödül, kimya, fizik ve bilgisayar bilimi arasında oluşmuş bir "bilimsel köprü" düğümüdür.

Benzer şekilde 2001 Nobel Fizik Ödülü (Eric Cornell, Wolfgang Ketterle, Carl Wieman) Bose-Einstein yoğunlaşması üzerine yapılan çalışmalarla ilişkilidir. Bu keşif, farklı laboratuvarlarda paralel olarak gelişmiş ve daha sonra birleşerek büyük bir bilimsel kırılma noktası oluşturmuştur. Bu tür paralel gelişmeler, ağ teorisi açısından "eşzamanlı küresel keşifler" olarak modellenebilir.

Nobel ödülleri üzerinden oluşturulan bilgi ağları, bilimsel yeniliğin nasıl yayıldığını da gösterir. Bir keşif, başlangıçta küçük bir düğümken zamanla genişleyen bir etki alanı oluşturur. Bu süreç "bilgi difüzyonu" olarak adlandırılır. Örneğin CRISPR gen düzenleme teknolojisi, başlangıçta dar bir biyoloji alanında ortaya çıkmışken kısa sürede genetik, tıp ve biyoteknoloji alanlarına yayılmıştır. Eğer Nobel süreci devam eden bir ağ olarak ele alınırsa, CRISPR düğümünün hızla büyüyen bir merkez olduğu görülür.

Nobel ağı aynı zamanda "bilimsel etki yoğunluğu"nu ölçmek için de kullanılabilir. Bazı Nobel ödülleri yalnızca bir keşfi değil, bütün bir araştırma alanını temsil eder. Örneğin Albert Einstein'ın çalışmaları, sadece fizik içinde değil, modern teknolojinin birçok alanında etkili olan geniş bir bilgi ağı oluşturur. Bu nedenle Einstein düğümü, yalnızca bir kişi değil, çok katmanlı bir bilimsel etki alanıdır.

Bilgi grafı yaklaşımı, Nobel Ödülleri arasındaki görünmez bağlantıları da ortaya çıkarır. Örneğin farklı yıllarda ödül almış bilim insanlarının çalışmaları arasında doğrudan atıf olmasa bile, ortak teorik çerçeveler veya yöntemsel benzerlikler bulunabilir. Bu tür ilişkiler, klasik bibliyometrik analizlerde görünmezken graf analizde ortaya çıkar.

Ayrıca Nobel Ödülleri, bilimsel disiplinlerin evrimini anlamak için de önemli bir araçtır. Fizik, kimya, tıp ve ekonomi gibi alanlar zaman içinde birbirleriyle giderek daha fazla bağlantılı hale

gelmiştir. Bu durum, bilimin izole disiplinlerden çok disiplinler arası bir ağ yapısına dönüştüğünü gösterir.

Sonuç olarak Nobel Ödülleri ve bilgi ağları birlikte ele alındığında, bilimsel ilerlemenin yalnızca bireysel başarılarla değil, karmaşık bir ilişki ağıyla gerçekleştiği görülür. Bilgi grafı yaklaşımı bu ağı görünür hale getirerek bilimin gerçek yapısını daha derinlemesine anlamamıza olanak tanır.

8.4 Protein Etkileşim Ağları

Protein etkileşim ağları (Protein-Protein Interaction Networks), biyolojinin en karmaşık ve en önemli bilgi yapılarından biridir. Hücre içinde gerçekleşen tüm biyolojik süreçler, proteinlerin birbirleriyle kurduğu etkileşimler sayesinde mümkün olur. Bu etkileşimler, doğal olarak bir ağ yapısı oluşturur ve bu ağın anlaşılması, yaşamın temel mekanizmalarının anlaşılması anlamına gelir.

Bir hücreyi düşündüğümüzde, binlerce protein sürekli olarak birbirleriyle iletişim halindedir. Bu proteinler tek başına işlevsel değildir; çoğu zaman bir başka proteinle bağlanarak bir işlevi yerine getirir. Örneğin bir sinyal iletim yolu, bir proteinin diğerini aktive etmesiyle başlar ve zincirleme reaksiyonlarla devam eder. Bu süreçler graf yapılarla temsil edildiğinde, hücrenin bir "biyolojik bilgi grafı" olduğu ortaya çıkar.

Protein etkileşim ağlarında her düğüm bir proteini temsil ederken, her bağlantı iki protein arasındaki fiziksel veya fonksiyonel etkileşimi gösterir. Örneğin "Protein A → bağlanır → Protein B" ilişkisi, hücre içinde gerçekleşen bir biyokimyasal etkileşimi ifade eder. Bu tür ilişkiler birleştiğinde oldukça karmaşık ağ yapıları oluşur.

Bu ağların en önemli özelliklerinden biri ölçekten bağımsız (scale-free) yapıya sahip olmalarıdır. Yani bazı proteinler çok sayıda etkileşime sahipken, çoğu protein az sayıda bağlantıya sahiptir. Bu durum, biyolojik sistemlerin dayanıklılığını ve esnekliğini açıklar. Az sayıda "hub protein" (merkez protein) tüm sistemin işleyişinde kritik rol oynar.

Örneğin p53 proteini, hücre döngüsünü düzenleyen en önemli proteinlerden biridir ve çok sayıda farklı proteinle etkileşim halindedir. Bu nedenle p53, protein etkileşim ağında merkezi bir düğüm olarak kabul edilir. Eğer bu protein işlevini kaybederse, kanser gibi ciddi hastalıklar ortaya çıkabilir.

Protein etkileşim ağları, ilaç geliştirme süreçlerinde de kritik rol oynar. Bir ilacın etkisi yalnızca tek bir proteine değil, genellikle bir protein ağına etki eder. Bu nedenle modern ilaç tasarımı, tek hedefli değil ağ tabanlı bir yaklaşım gerektirir. Örneğin bir ilaç, yalnızca bir proteini inhibe etmekle kalmaz, aynı zamanda bağlı olduğu diğer proteinlerin davranışını da değiştirir.

Bilgi grafı yaklaşımı, protein etkileşimlerini sistematik olarak modellemeyi mümkün kılar. Büyük biyolojik veri tabanları (örneğin STRING veritabanı) protein-protein ilişkilerini graf yapılar halinde sunar. Bu sayede araştırmacılar, hücre içindeki karmaşık etkileşimleri görselleştirebilir ve analiz edebilir.

Bir örnek üzerinden düşünelim: “Protein A → aktive eder → Protein B”, “Protein B → baskılar → Protein C”, “Protein C → düzenler → hücre bölünmesi”. Bu zincir, hücresel bir süreci temsil eder ve graf üzerinde analiz edildiğinde biyolojik mekanizmanın tamamı görülebilir.

Ayrıca protein etkileşim ağları, hastalıkların moleküler düzeyde anlaşılmasına da yardımcı olur. Örneğin Alzheimer hastalığında belirli proteinlerin yanlış katlanması ve diğer proteinlerle anormal etkileşimler kurması, hastalığın ilerlemesini etkiler. Graf analiz bu tür patolojik ağları ortaya çıkarabilir.

Sonuç olarak protein etkileşim ağları, yaşamın temelini oluşturan biyolojik süreçlerin ilişkisel doğasını ortaya koyar. Bilgi grafı yaklaşımı sayesinde bu karmaşık sistemler anlaşılabilir, analiz edilebilir ve hatta müdahale edilebilir hale gelir. Bu da modern biyolojiyi yalnızca bir moleküler bilim olmaktan çıkarıp, aynı zamanda büyük ölçekli bir ağ bilimi haline getirir.

8.5 Genetik ve Biyoinformatik

Genetik ve biyoinformatik, yaşamın en temel düzeyde nasıl işlediğini anlamaya çalışan iki disiplin olarak, bilgi grafı yaklaşımıyla doğal bir birleşim alanı oluşturur. Genetik, DNA, genler ve kalıtım mekanizmalarıyla ilgilenirken; biyoinformatik bu biyolojik verileri hesaplamalı yöntemlerle analiz eder. Ancak bu verilerin doğası, yalnızca sayısal değil, aynı zamanda derin biçimde ilişkisel olduğu için graf tabanlı modeller bu alanlarda giderek daha merkezi hale gelmiştir.

Bir organizmanın genetik yapısı, aslında devasa bir bilgi ağıdır. DNA üzerindeki her gen, diğer genlerle, proteinlerle, biyokimyasal yollarla ve çevresel faktörlerle etkileşim içindedir. Bu ilişkiler düz bir liste ya da tablo ile açıklanamaz; bunun yerine çok katmanlı bir ağ yapısı gerektirir. Örneğin “Gen A → düzenler → Gen B” veya “Gen C → kodlar → Protein X” gibi ilişkiler, biyolojik sistemin temel grafsel yapı taşlarıdır.

Biyoinformatik, bu karmaşık veriyi anlamlandırmak için bilgi grafı yaklaşımını giderek daha fazla kullanmaktadır. Özellikle genetik varyasyonlar, mutasyonlar ve hastalık ilişkileri graf yapılarla modellenebilir. Örneğin “BRCA1 geni → ilişkilidir → meme kanseri” veya “CFTR mutasyonu → neden olur → kistik fibrozis” gibi bağlantılar, hastalık-gen ilişkilerinin ağ tabanlı temsilini sağlar.

Genetik ağların en önemli özelliklerinden biri çok katmanlı yapıya sahip olmalarıdır. Bir gen yalnızca başka bir genle değil, aynı zamanda proteinlerle, metabolik yollarla ve hücresel süreçlerle de etkileşim içindedir. Bu nedenle biyolojik sistemler tek katmanlı değil, üst üste binen çoklu ağlardan oluşur. Bu yapı, “çok katmanlı biyolojik graflar” olarak modellenir.

Örneğin bir hücrede “Gen A → ifade eder → Protein A”, “Protein A → inhibe eder → Enzim B”, “Enzim B → etkiler → Metabolik yol C” zinciri, biyolojik sürecin grafsal bir temsilidir. Bu zincirler birleştiğinde hücrenin işleyişini açıklayan büyük bir sistem ortaya çıkar.

Genetik verilerin graf yapılarla modellenmesi, özellikle hastalık araştırmalarında büyük bir devrim yaratmıştır. Karmaşık hastalıklar genellikle tek bir genle değil, birden fazla gen ve çevresel faktörün etkileşimiyle ortaya çıkar. Örneğin diyabet, yalnızca bir genetik faktör değil,

çok sayıda genin ve metabolik yolun etkileşimiyle oluşan bir ağ hastalığıdır. Bu nedenle klasik tek-gen yaklaşımı yetersiz kalır.

Biyoinformatikte bilgi grafı, ilaç hedeflerinin belirlenmesinde de kullanılır. Bir ilacın hangi proteinlere, hangi genlere ve hangi biyolojik yollarla etki ettiğini anlamak için graf tabanlı analizler yapılır. Örneğin "İlaç X → inhibe eder → Protein Y → etkiler → Hücre bölünmesi" ilişkisi, potansiyel tedavi mekanizmalarını ortaya çıkarır.

Ayrıca genetik ağlar, evrimsel biyoloji çalışmalarında da kullanılır. Farklı türler arasındaki genetik benzerlikler graf yapılarla karşılaştırılarak evrimsel ilişkiler analiz edilir. Örneğin insan ve şempanze genomları arasındaki benzer gen ağları, ortak evrimsel geçmişi gösterir.

Modern biyoinformatik sistemlerinde yapay zekâ ve graf sinir ağları (Graf Neural Networks) da kullanılmaktadır. Bu modeller, genetik ağlar üzerinde öğrenme yaparak yeni biyolojik ilişkileri tahmin edebilir. Örneğin daha önce bilinmeyen bir genin hangi hastalıkla ilişkili olabileceği graf öğrenme yöntemleriyle tahmin edilebilir.

Sonuç olarak genetik ve biyoinformatik, yaşamın temelini oluşturan bilgiyi grafsel bir sistem olarak ele alır. Bu yaklaşım, biyolojiyi yalnızca moleküler düzeyde değil, aynı zamanda ilişkisel ve sistemik bir düzeyde anlamamıza olanak tanır. Bilgi grafı burada yalnızca bir veri modeli değil, yaşamın kendisini temsil eden bir düşünme biçimi haline gelir.

8.6 Fikirlerin Evrimi

Fikirlerin evrimi, insan düşüncesinin tarih boyunca nasıl değiştiğini ve geliştiğini anlamaya çalışan disiplinler arası bir konudur. Bu süreç yalnızca bireysel düşüncelerle değil, aynı zamanda toplumsal, kültürel ve bilimsel etkileşimlerle şekillenir. Bilgi grafı yaklaşımı, fikirlerin bu evrimsel sürecini modellemek için güçlü bir araç sunar çünkü fikirler doğası gereği ilişkisel yapılardır.

Her fikir, başka fikirlerle bağlantılıdır. Hiçbir düşünce tamamen izole değildir; her biri önceki bilgi birikiminin üzerine inşa edilir. Örneğin modern fizik, Newton mekaniği, Einstein'ın görelilik teorisi ve kuantum mekaniği arasındaki ilişkilerle şekillenmiştir. Bu fikirler bir graf yapıda temsil edildiğinde, bilimsel düşüncenin evrimsel haritası ortaya çıkar.

Fikirlerin evriminde üç temel mekanizma vardır: birleşme, ayrışma ve mutasyon. Birleşme, farklı fikirlerin bir araya gelerek yeni bir fikir oluşturmasıdır. Ayrışma, tek bir fikrin farklı yönlere bölünmesidir. Mutasyon ise mevcut bir fikrin küçük değişikliklerle yeni bir forma dönüşmesidir. Bu mekanizmalar, biyolojik evrimdeki süreçlere oldukça benzer ve bu nedenle "kültürel evrim" kavramı sıkça kullanılır.

Örneğin yapay zekâ fikri, matematik, nörobilim ve bilgisayar biliminin birleşimiyle ortaya çıkmıştır. Bu üç alan arasındaki bağlantılar graf üzerinde gösterildiğinde, yapay zekânın tek bir disiplinden değil, bir ağ yapısından doğduğu görülür. "Sinir ağı → ilham alır → biyolojik sinir sistemi" veya "istatistik → temel oluşturur → makine öğrenmesi" gibi ilişkiler bu evrimi açıklar.

Fikirlerin evrimi aynı zamanda bilgi yayılımı süreçleriyle de ilgilidir. Bir fikir ortaya çıktıktan sonra akademik yayınlar, kitaplar, konferanslar ve dijital platformlar aracılığıyla yayılır. Bu yayılım süreci graf modelleme ile analiz edilebilir. Örneğin "Darwin'in evrim teorisi → etkiler → modern biyoloji" ilişkisi, fikirlerin uzun vadeli etkisini gösterir.

Bilgi grafı, fikirler arasındaki görünmeyen bağlantıları da ortaya çıkarabilir. Farklı alanlarda bağımsız olarak ortaya çıkan benzer fikirler, graf analizle ilişkilendirilebilir. Bu durum "paralel keşif" olarak adlandırılır. Örneğin farklı bilim insanlarının aynı dönemde benzer teorilere ulaşması, ağ içinde yakın düğümler olarak görülebilir.

Fikirlerin evriminde merkezilik ve köprü düğümler de önemli rol oynar. Bazı fikirler çok sayıda diğer fikre bağlanarak merkezi bir konum kazanır. Bazı fikirler ise farklı alanlar arasında köprü görevi görür. Örneğin "bilgi teorisi" hem matematik hem de iletişim bilimi arasında bir köprü fikirdir.

Modern dijital çağda fikirlerin evrimi çok daha hızlı hale gelmiştir. Sosyal medya, akademik yayınlar ve açık veri platformları fikirlerin yayılma hızını artırmıştır. Bu durum aynı zamanda fikirlerin daha hızlı mutasyona uğramasına da neden olur. Graf tabanlı analizler, bu hızlı değişimi anlamak için kritik bir araçtır.

Sonuç olarak fikirlerin evrimi, insan düşüncesinin tarihsel ve yapısal bir ağ içinde nasıl geliştiğini gösterir. Bilgi grafı yaklaşımı, fikirleri statik varlıklar olarak değil, sürekli değişen ve birbirine bağlı bir sistem olarak görmemizi sağlar. Bu perspektif, düşüncenin kendisini bir "yaşayan ağ" olarak yeniden tanımlar.

8.7 Bilim İnsanları Arasındaki Gizli Bağlantılar

Bilim, yüzeyde bireysel keşiflerin toplamı gibi görünür: bir isim, bir makale, bir buluş, bir ödül. Ancak bu görünür katmanın altında çok daha derin bir yapı vardır. Bilimsel üretim, aslında birbirine görünür ya da görünmez bağlarla bağlı insanlardan oluşan dev bir ağ içinde gerçekleşir. Bu ağın düğümleri bilim insanlarıdır; bağlantılar ise ortak çalışmalar, atıflar, aynı fikir gelenekleri, aynı kurumlarda bulunma, aynı öğrencileri yetiştirme ya da dolaylı entelektüel etkiler gibi çok çeşitli ilişkilerdir. Bilgi grafı yaklaşımı, bu görünmeyen yapıyı görünür hale getiren en güçlü araçlardan biridir.

Bilim insanları arasındaki ilişkiler çoğu zaman doğrudan iş birliği şeklinde değildir. Bir araştırmacı başka bir araştırmacının makalesine atıf yapabilir, ancak hiç tanışmamış olabilirler. Yine de bu iki kişi aynı bilgi ağının içinde birbirine bağlıdır. Bu tür dolaylı bağlantılar, bilimsel bilginin en önemli taşıyıcılarından biridir. Örneğin Albert Einstein'ın çalışmaları, doğrudan tanışmadığı birçok fizikçi tarafından geliştirilmiş ve genişletilmiştir. Bu durum, bilimsel etkinin kişisel ilişkilerden çok daha geniş bir ağ üzerinden yayıldığını gösterir.

Gizli bağlantılar kavramı burada kritik bir rol oynar. Gizli bağlantılar, yüzeyde görünmeyen ancak veri analizleriyle ortaya çıkarılabilen ilişkilerdir. Örneğin iki bilim insanı hiçbir zaman birlikte çalışmamış olabilir, ancak aynı teorik çerçevede çalışmış, benzer makalelere atıf yapmış

ya da aynı araştırma sorularını farklı açılardan ele almış olabilir. Bu tür ilişkiler, graf analiz yöntemleriyle ortaya çıkarılabilir ve bilimsel alanın gerçek yapısını anlamamıza yardımcı olur.

Bilim insanları arasındaki ağ yapısı genellikle "küçük dünya ağı" özellikleri gösterir. Bu, herhangi iki bilim insanının birkaç adımda birbirine bağlanabileceği anlamına gelir. Örneğin bir fizikçi ile bir biyolog arasında doğrudan bir ilişki olmayabilir, ancak birkaç ara araştırmacı üzerinden dolaylı bir bağlantı kurulabilir. Bu yapı, bilimin neden bu kadar hızlı ilerleyebildiğini de açıklar: bilgi ağ içinde çok hızlı dolaşır.

Bir diğer önemli kavram "köprü bilim insanları"dır. Bu kişiler farklı disiplinler arasında bağlantı kuran araştırmacılardır. Örneğin bir kişi hem bilgisayar bilimi hem de biyoloji alanında çalışıyorsa, bu iki alan arasında bir köprü oluşturur. Bu tür köprü düğümler, bilimsel inovasyonun en önemli kaynaklarından biridir çünkü farklı bilgi alanlarının birleşmesini sağlarlar. Yapay zekâ alanının gelişimi de tam olarak böyle köprülerin sonucudur.

Bilim insanları arasındaki gizli bağlantılar aynı zamanda akademik soy ağaçları (academic genealogy) üzerinden de incelenebilir. Bir öğrencinin hocasıyla, hocasının hocasıyla ve onların akademik atalarıyla olan ilişkileri, bilimsel bilginin kuşaklar arası aktarımını gösterir. Bu yapı, bilimsel düşüncenin yalnızca yatay değil, dikey bir ağ içinde de evrildiğini ortaya koyar.

Örneğin bir matematikçinin doktora hocası, onun fikir dünyasını derinden etkiler. Bu etki bazen doğrudan öğretilen konularla, bazen de düşünme biçimiyle ilgilidir. Bu nedenle bilimsel soy ağaçları yalnızca akademik unvan zincirleri değil, aynı zamanda entelektüel etki zincirleridir.

Gizli bağlantılar ayrıca bilimsel rekabeti de içerir. Aynı problemi çözmeye çalışan farklı araştırma grupları, birbirlerinin çalışmalarını takip eder, atıf yapar ve bazen dolaylı olarak birbirlerinin çalışmalarını şekillendirir. Bu rekabet, bilimsel ilerlemeyi hızlandıran önemli bir dinamik oluşturur. Graf analizler bu rekabet ağlarını da görünür hale getirebilir.

Bilim insanları arasındaki ilişkiler zaman içinde değişir. Bir araştırmacı bir dönem merkezi bir konumda olabilirken, başka bir dönemde daha periferik bir role geçebilir. Bu dinamik yapı, bilimin statik değil, sürekli değişen bir ağ olduğunu gösterir. Özellikle büyük bilimsel kırılma noktaları (örneğin kuantum mekaniği veya genetik devrim) sırasında ağ yapısı hızla yeniden şekillenir.

Dijital çağda bu gizli bağlantılar çok daha görünür hale gelmiştir. Akademik veri tabanları, atıf indeksleri ve açık erişim platformları sayesinde bilim insanları arasındaki ilişkiler büyük veri analiziyle incelenebilir hale gelmiştir. Örneğin bir araştırmacının kimlere atıf yaptığı, kimlerle ortak makale yazdığı ve hangi alanlarda yoğunlaştığı graf olarak görselleştirilebilir.

Bu analizler yalnızca geçmişi anlamak için değil, geleceği tahmin etmek için de kullanılabilir. Örneğin hangi araştırmacıların gelecekte önemli bir bilimsel atılım yapabileceği, onların ağ içindeki konumlarına bakılarak tahmin edilebilir. Bu, "bilimsel potansiyel analizi" olarak adlandırılabilir.

Sonuç olarak bilim insanları arasındaki gizli bağlantılar, bilimin görünmeyen mimarisini oluşturur. Bu mimari, bireylerden çok ilişkiler üzerine kuruludur. Bilgi grafı yaklaşımı, bu ilişkileri görünür hale getirerek bilimi yalnızca bir keşifler dizisi olarak değil, yaşayan, evrilen ve sürekli yeniden şekillenen bir insan ağı olarak anlamamıza olanak tanır.

BÖLÜM 9 — KODLAMA VE PROJELER

Bu bölüm, kitabın teorik omurgasını somut uygulamalara dönüştüren laboratuvar kısmıdır. Buraya kadar anlatılan kavramlar—graf düşüncesi, bilgi grafı mimarisi, ontolojiler, varlıklar ve ilişkiler—artık kod düzeyinde karşılık bulur. Amaç yalnızca bir graf veritabanını kullanmayı öğretmek değildir; aynı zamanda okuyucunun kendi "küçük bilgi sistemi"ni kurabilmesini sağlamaktır. Bu nedenle bölüm, basit veri yapılarından başlayarak profesyonel graf veri tabanı sistemlerine ve yapay zekâ entegrasyonuna kadar genişleyen bir uygulama hattı içerir.

Kodlama tarafı, graf düşüncenin en kritik yönünü görünür kılar: ilişkiler aslında soyut değil, doğrudan temsil edilebilir yapılardır. Bir düğüm ve bir bağlantı, yalnızca teorik kavramlar değil, programlanabilir nesnelerdir. Python gibi diller bu yapıyı oldukça doğal biçimde modellemeye izin verir. Sözlükler, listeler ve sınıflar kullanılarak küçük ölçekli graf sistemleri kurulabilir ve bu sistemler üzerinden temel ağ işlemleri test edilebilir.

NetworkX gibi kütüphaneler, bu düşünceyi bir adım ileri taşır. Kullanıcı, birkaç satır kodla ağ oluşturabilir, merkezilik ölçümleri yapabilir ve topluluk yapılarını analiz edebilir. Bu tür araçlar, graf düşüncenin sezgisel olarak anlaşılmasını sağlar. Özellikle sosyal ağ örnekleri, kavramın günlük hayata nasıl karşılık geldiğini gösterir.

Daha ileri seviyede Neo4j gibi graf veritabanları devreye girer. Bu sistemler, büyük ölçekli ilişkisel veriyi saklamak ve sorgulamak için geliştirilmiştir. Cypher gibi sorgu dilleri sayesinde kullanıcı, "kim kiminle bağlantılı?", "hangi yol üzerinden bağlanıyor?" gibi soruları doğrudan graf üzerinde sorabilir. Bu yaklaşım, klasik SQL mantığından farklı olarak ilişkilerin doğrudan sorgulanmasına olanak tanır.

Son yıllarda ortaya çıkan Kùzu gibi gömülü graf veritabanları ise bu yapıyı daha hafif ve yerel hale getirmiştir. Bu sistemler özellikle araştırmacılar ve bireysel geliştiriciler için tasarlanmıştır. Python entegrasyonu sayesinde, veri dosyaları doğrudan graf yapıya dönüştürülebilir ve yerel bilgisayar üzerinde analiz edilebilir. Bu yaklaşım, büyük sistemlere bağımlılığı azaltır ve "kişisel bilgi grafı laboratuvarı" fikrini mümkün kılar.

Bölümün ilerleyen kısımlarında veri çıkarımı, PDF ve web kaynaklarından bilgi toplama, Wikipedia üzerinden otomatik ağ oluşturma gibi uygulamalar yer alır. Bu süreçler, yapılandırılmamış verinin nasıl graf formuna dönüştürülebileceğini gösterir. Aynı zamanda büyük dil modelleri (LLM) ile bilgi grafı entegrasyonu, modern yapay zekâ sistemlerinin nasıl çalıştığını anlamak için kritik bir örnek sunar.

Bu bölüm yalnızca teknik bir rehber değil, aynı zamanda bir düşünme biçimi değişimidir. Okuyucu, satır ve sütunlardan oluşan klasik veri dünyasından çıkarak, ilişkilerle düşünen bir

sistem tasarımcısına dönüşür. Kod, burada yalnızca bir araçtır; asıl amaç, dünyayı grafler üzerinden modelleme yeteneği kazanmaktır.

9.1 Python ile Graf Veri Yapıları

Graf veri yapıları, bilgi grafı düşüncesinin en temel programlama karşılığıdır. Bir sistemdeki varlıkları "düğüm" ve bu varlıklar arasındaki ilişkileri "bağlantı " olarak temsil etmek, karmaşık yapıları anlaşılır ve işlenebilir hale getirir. Python, bu yapıyı kurmak için özellikle uygun bir dildir çünkü esnek veri tipleri ve okunabilir sözdizimi sayesinde graf modelleme doğrudan ve sezgisel biçimde yapılabilir.

Düğüm ve Bağlantı Mantığı

Graf yapısının en temel bileşeni düğümdür. Düğüm, bir kişi, bir şehir, bir makale ya da bir kavram olabilir. Bağlantı ise bu düğümler arasındaki ilişkiyi temsil eder. Örneğin:

- "Ali → çalışır → Üniversite"
- "Makine Öğrenmesi → alt alanıdır → Yapay Zekâ"

Bu tür ilişkiler, klasik tablo yapılarında çok sayıda JOIN işlemi gerektirirken, graf yapısında doğrudan temsil edilir. Python'da bu yapı genellikle sözlükler ve listeler kullanılarak modellenir.

Komşuluk Listesi Yaklaşımı

Graf yapılarında en yaygın temsil yöntemlerinden biri komşuluk listesi (adjacency list) yaklaşımıdır. Bu yöntemde her düğüm, kendisine bağlı diğer düğümlerin bir listesi ile temsil edilir.

Örneğin:

```
graf = {
   "Ali": ["Üniversite", "Şirket"],
   "Üniversite": ["Bölüm1", "Bölüm2"],
   "Şirket": ["ProjeA"]
}
```

Bu yapı, her düğümün doğrudan bağlantılarını saklar. Büyük veri setlerinde bu yöntem hem bellek açısından verimlidir hem de hızlı erişim sağlar.

Basit Graf Sınıfı Yazımı

Graf yapısını daha sistematik hale getirmek için Python'da bir sınıf (class) tanımlamak mümkündür. Bu yaklaşım, düğüm ekleme, bağlantı oluşturma ve sorgulama işlemlerini daha düzenli hale getirir.

Örneğin temel bir graf sınıfı şu mantıkla tasarlanabilir:

- Düğümleri saklayan bir yapı

- Bağlantıları ekleyen fonksiyonlar
- Bağlantıları döndüren metotlar

Bu yapı sayesinde graf yalnızca bir veri yapısı değil, aynı zamanda davranışları olan bir nesne haline gelir. Bu, nesne yönelimli programlamanın (object-oriented programming) graf sistemlerine uygulanmasıdır.

Python Sözlükleri ile Ağ Modelleme

Python sözlükleri, graf yapıları için doğal bir modelleme aracıdır. Çünkü her anahtar bir düğümü, değer ise bu düğümün bağlantılarını temsil edebilir. Bu yaklaşım özellikle küçük ve orta ölçekli ağlarda oldukça etkilidir.

Örneğin bir akademik iş birliği ağı şu şekilde temsil edilebilir:

```
network = {
   "Alice": ["Bob", "Carol"],
   "Bob": ["Alice", "David"],
   "Carol": ["Alice"],
   "David": ["Bob"]
}
```

Bu yapı, sosyal ağlar, makale atıf ağları veya ürün ilişkileri gibi birçok farklı alana kolayca uyarlanabilir. Sözlük yapısının esnekliği, graf modellemenin temel avantajlarından biridir.

İlk Gezinme Örnekleri

Graf yapılarında en önemli işlemlerden biri ağ üzerinde gezinmedir. Traversal, bir düğümden başlayarak bağlı tüm düğümlerin keşfedilmesini sağlar. En yaygın iki yöntem vardır: derinlik öncelikli arama (DFS) ve genişlik öncelikli arama (BFS).

Basit bir DFS mantığı şu şekilde düşünülebilir:

- Bir düğümden başla
- Komşularına git
- Ziyaret edilmemiş düğümleri tekrar gez

Bu süreç, graf içindeki tüm bağlantıların sistematik olarak keşfedilmesini sağlar. BFS ise katman katman ilerleyerek en yakın komşulardan uzaklara doğru genişler.

Örneğin bir sosyal ağda “Ali”den başlayarak tüm bağlantıları keşfetmek, onun arkadaşlarını, arkadaşlarının arkadaşlarını ve daha uzak bağlantıları ortaya çıkarır. Bu tür işlemler, bilgi grafı sistemlerinin temel analiz araçlarıdır.

Sonuç

Python ile graf veri yapıları oluşturmak, yalnızca bir programlama egzersizi değildir; aynı zamanda dünyayı ilişkiler üzerinden modelleme pratiğidir. Düğüm ve bağlantı mantığı, komşuluk listesi yaklaşımı, sözlük tabanlı modelleme ve gezinme algoritmaları bir araya geldiğinde, karmaşık sistemler basit ve anlaşılır yapılara dönüşür. Bu da bilgi grafı düşüncesinin en temel programatik karşılığıdır.

9.2 NetworkX Kullanımı

NetworkX, Python ekosisteminde graf (graf) teorisini pratik uygulamaya dönüştüren en önemli kütüphanelerden biridir. Basit ama güçlü yapısı sayesinde, düğümler ve bağlantılar ile temsil edilen karmaşık ağları oluşturmak, analiz etmek ve görselleştirmek mümkün hale gelir. Bu bölümde NetworkX'in kurulumu, temel ağ oluşturma süreçleri ve graf analizinin temel kavramları adım adım ele alınacaktır.

NetworkX Kurulumu

NetworkX, Python paket yöneticisi pip aracılığıyla kolayca kurulabilir:

pip install networkx

Kurulumdan sonra genellikle görselleştirme için matplotlib kütüphanesi de eklenir:

pip install matplotlib

NetworkX'in en önemli avantajı, kullanıcıya düşük seviyeli veri yapılarıyla uğraşmadan doğrudan graf nesneleriyle çalışma imkânı vermesidir. Bu, özellikle bilgi grafı çalışmalarında büyük bir hız kazandırır.

İlk Ağın Oluşturulması

NetworkX ile ilk grafi oluşturmak oldukça sezgiseldir. Öncelikle bir graf nesnesi tanımlanır, ardından düğümler ve bağlantılar eklenir.

```
import networkx as nx

G = nx.Graf()

G.add_node("Ali")
G.add_node("Ayşe")

G.add_edge("Ali", "Ayşe")
```

Bu basit örnek, iki kişi arasında bir ilişki kurar. Ancak NetworkX'in gücü burada başlar: yüzlerce hatta milyonlarca düğüm ve ilişkiyi aynı yapı içinde yönetebilir.

Düğümler yalnızca isimlerden ibaret değildir; her düğüme özellik eklemek mümkündür:

```
G.add_node("Ali", role="researcher", age=35)
```

Bu özellik, bilgi grafı yaklaşımında çok önemlidir çünkü varlıklar yalnızca kimlik değil, aynı zamanda anlam taşır.

Merkeziyet Ölçümleri

Graf analizinin en kritik kavramlarından biri merkeziyettir (centrality). Merkeziyet, bir düğümün ağ içindeki önemini ölçer. NetworkX bu konuda çeşitli metrikler sunar:

1. Derece Merkeziyeti
Bir düğümün kaç bağlantıya sahip olduğunu gösterir.

nx.degree_centrality(G)

2. Aradalık Merkeziyeti
Bir düğümün diğer düğümler arasındaki en kısa yollar üzerinde ne kadar yer aldığını ölçer. Bu, "köprü düğümler"i bulmak için önemlidir.

nx.betweenness_centrality(G)

3. Yakınlık Merkeziyeti
Bir düğümün diğer tüm düğümlere ne kadar yakın olduğunu ölçer.

Bu metrikler, özellikle sosyal ağlar ve bilgi grafı sistemlerinde hangi varlıkların daha kritik olduğunu anlamak için kullanılır. Örneğin bir akademik ağda en çok atıf alan araştırmacılar genellikle yüksek merkeziyete sahiptir.

Topluluk Analizi

Gerçek dünyadaki ağlar genellikle rassal değil, kümelenmiş yapılardır. İnsanlar arkadaş grupları oluşturur, makaleler belirli konular etrafında kümelenir, web sayfaları belirli temalarda yoğunlaşır. Bu kümeleri tespit etmek için topluluk analizi kullanılır. NetworkX, bu amaçla çeşitli algoritmalar sunar. En yaygın yöntemlerden biri "modülerlik tabanlı kümeleme"dir.

```
from networkx.algorithms import community
communities = community.greedy_modularity_communities(G)
```

Bu analiz sayesinde ağ içinde birbirine sıkı bağlı alt gruplar ortaya çıkar. Bilgi grafı açısından bu, kavramsal alanların otomatik keşfi anlamına gelir. Örneğin biyoloji ile ilgili makaleler bir küme oluştururken, yapay zekâ ile ilgili makaleler başka bir küme oluşturabilir.

Görselleştirme Teknikleri

NetworkX'in en güçlü yönlerinden biri matplotlib ile birlikte kullanılarak graflarin görselleştirilmesidir. Basit bir görselleştirme şu şekilde yapılabilir:

```
import matplotlib.pyplot as plt


nx.draw(G, with_labels=True)
plt.show()
```

Ancak daha gelişmiş görselleştirmelerde düğüm boyutu, renk ve konum gibi özellikler merkeziyet veya topluluk bilgisine göre ayarlanabilir. Örneğin merkezi düğümler daha büyük gösterilebilir, farklı topluluklar farklı renklerle temsil edilebilir.

Bu görselleştirme teknikleri, karmaşık ağların sezgisel olarak anlaşılmasını sağlar. Özellikle bilgi grafı sistemlerinde görselleştirme, veri analizi kadar önemlidir çünkü ilişkisel yapıların insan zihninde kavranmasını kolaylaştırır.

Sosyal Ağ Örneği

NetworkX'in en yaygın kullanım alanlarından biri sosyal ağ analizidir. Basit bir sosyal ağ modeli düşünelim:

```
social = nx.Graf()


social.add_edges_from([
   ("Ali", "Ayşe"),
   ("Ali", "Mehmet"),
   ("Ayşe", "Zeynep"),
   ("Mehmet", "Zeynep"),
   ("Zeynep", "Elif")
])
```

Bu ağ üzerinde merkeziyet analizi yapıldığında, bazı bireylerin daha merkezi konumda olduğu görülür. Örneğin "Zeynep" hem Ali hem Mehmet ile bağlantılı olduğu için köprü bir düğüm olabilir.

Topluluk analizi yapıldığında ise bu küçük ağ bile iki veya daha fazla alt gruba ayrılabilir. Bu tür analizler, sosyal medya platformlarından akademik iş birliklerine kadar birçok alanda kullanılır

Sonuç

NetworkX, graf düşüncesini kod seviyesinde somutlaştıran en güçlü araçlardan biridir. Kurulumu kolay, kullanımı sezgisel ve analitik gücü yüksektir. Düğüm ve bağlantı mantığından başlayarak merkeziyet, topluluk analizi ve görselleştirme tekniklerine kadar geniş bir araç seti sunar. Bu sayede yalnızca veri analizi değil, aynı zamanda ilişkisel düşünme biçimi de geliştirilir.

Özellikle bilgi grafı yaklaşımında NetworkX, küçük ve orta ölçekli sistemler için ideal bir laboratuvar ortamı sağlar. Bu bölümde öğrenilen kavramlar, daha sonra Neo4j ve Kùzu gibi profesyonel sistemlere geçiş için sağlam bir temel oluşturur.

9.3 Neo4j Kurulumu

Neo4j, graf veri tabanı dünyasının en yaygın kullanılan ve en olgun sistemlerinden biridir. İlişkisel verileri düğümler ve ilişkiler şeklinde saklayarak, özellikle yoğun bağlantı içeren veri yapılarında klasik ilişkisel veri tabanlarına kıyasla çok daha doğal ve hızlı bir model sunar. Bu

bölümde Neo4j'in kurulumu, temel kullanım adımları ve ilk graf veri tabanı deneyimi adım adım ele alınacaktır.

Neo4j Desktop Kurulumu

Neo4j'i kullanmaya başlamak için en kolay yöntem Neo4j Desktop uygulamasını kurmaktır. Neo4j Desktop, kullanıcıya yerel bilgisayarında graf veri tabanları oluşturma, yönetme ve test etme imkânı veren bir geliştirme ortamıdır.

Kurulum süreci oldukça basittir:

1. Neo4j'in resmi web sitesinden Neo4j Desktop indirilir.
2. Uygulama işletim sistemine göre yüklenir (Windows, macOS, Linux).
3. Kurulumdan sonra kullanıcı bir hesap oluşturur veya giriş yapar.

Neo4j Desktop'un en önemli avantajı, tüm graf veri tabanı projelerini tek bir arayüzden yönetebilmesidir. Ayrıca farklı projeler oluşturmak, sürümleri test etmek ve eklentileri kullanmak oldukça kolaydır. Şimdi bu kurulumun nasıl yapılacağını anlatalım:

1. Adım: Yükleme Dosyasını İndirme ve Başlatma

- Neo4j resmi web sitesinden işletim sisteminize (Windows, macOS veya Linux) uygun olan **Neo4j Desktop** yükleyicisini indirin.
- İndirilen .exe (Windows) veya .dmg (macOS) dosyasına çift tıklayarak kurulum sihirbazını başlatın.

2. Adım: Kullanıcı Seçimi (Windows için)

- Karşınıza gelen ilk ekranda, yazılımın bilgisayardaki hangi kullanıcılar için kurulacağını seçmeniz istenir:
    - *"Anyone who uses this computer (all users)"* (Tüm kullanıcılar)
    - *"Just for me"* (Sadece benim için)
- Genellikle **"Just for me"** seçeneği tavsiye edilir. Seçimi yapıp **Next (İleri)** butonuna tıklayın.

3. Adım: Kurulum Dizini Seçimi

- Bu ekranda Neo4j Desktop'ın yükleneceği klasör yolunu belirlersiniz (Örn: C:\Program Files\Neo4j Desktop).
- Varsayılan klasör yolunu değiştirmeden bırakabilir veya dilerseniz "Browse" diyerek başka bir konum seçebilirsiniz. **Install (Kur)** butonuna basarak yüklemeyi başlatın.

4. Adım: Lisans Sözleşmesi ve Kabul (İlk Açılış)

- Kurulum tamamlandıktan sonra programı çalıştırın. Karşınıza ilk olarak **"Software License Agreement"** (Yazılım Lisans Sözleşmesi) gelecektir.
- Şartları okuyup sayfanın altındaki **"I Accept" (Kabul Ediyorum)** butonuna tıklayarak ilerleyin.

5. Adım: Veri Depolama Konumu (Data Privacy & Storage)

- Bu ekranda Neo4j, projelerinizin ve graf veritabanı dosyalarınızın bilgisayarınızda hangi ana klasörde (Application Data Path) saklanacağını gösterir.
- Varsayılan yol genellikle kullanıcı klasörünüzün altındadır. Değiştirmek istemiyorsanız doğrudan **Confirm (Onayla)** butonuna tıklayın.

6. Adım: Aktivasyon Anahtarı ve Kayıt (Registration)

- Neo4j Desktop ücretsiz bir geliştirici lisansıyla gelir ancak aktivasyon gerektirir. Karşınıza çıkan formda:
    - Adınız, soyadınız, e-posta adresiniz ve şirket/kurum bilginizi girmeniz istenir.
- Bilgileri doldurup **Register (Kaydol)** dedikten sonra sistem size otomatik bir **Activation Key (Aktivasyon Anahtarı)** üretir.
- Bu anahtar bazen ekrandaki kutucuğa otomatik olarak kopyalanır, kopyalanmazsa tarayıcıda açılan sayfadaki anahtarı kopyalayıp bu alana yapıştırın ve **Activate** butonuna basın.

7. Adım: Neo4j Desktop Ana Ekranı (Dashboard)

- Aktivasyon tamamlandığında nihayet ana arayüze ulaşırsınız.
- Sol tarafta projelerinizi görebileceğiniz bir menü, orta kısımda ise varsayılan olarak gelen hazır bir **"Project"** ve içinde durdurulmuş halde bekleyen bir **"DBMS"** (Veritabanı Yönetim Sistemi) yer alır.

İlk Veritabanını Başlatma:

Kurulum bittikten sonra hemen test etmek için:

1. Ana ekrandaki varsayılan veritabanının yanındaki **Start** butonuna tıklayın.
2. Veritabanı aktif hale geldiğinde (yeşil ışık yandığında) **Open** butonuna basarak **Neo4j Browser** arayüzünü açabilir ve Cypher sorguları yazmaya başlayabilirsiniz.

İlk Graf Veri Tabanı

Kurulumdan sonra ilk adım yeni bir proje oluşturmaktır. Neo4j Desktop içinde “New Project” seçeneği ile bir proje oluşturulur ve bu projeye bir graf veri tabanı eklenir.

Her veri tabanı aslında bağımsız bir graf ortamıdır. Bu ortamda düğümler ve ilişkiler saklanır. Neo4j, veriyi klasik tablo mantığıyla değil, doğrudan graf yapısıyla depolar. İlk veri tabanı oluşturulduğunda genellikle sistem otomatik olarak bir "sandbox" ortamı sağlar. Bu ortamda kullanıcı test verileriyle çalışabilir[13].

Browser Arayüzü

Neo4j'in en güçlü araçlarından biri Neo4j Browser arayüzüdür. Bu arayüz, graf veri tabanı ile etkileşim kurmak için kullanılan görsel ve etkileşimli bir ortamdır.

Browser üzerinden:

- Cypher sorguları yazılabilir,
- Sonuçlar graf olarak görüntülenebilir,
- Düğümler ve ilişkiler görsel olarak incelenebilir.

Örneğin basit bir sorgu:

MATCH (n) RETURN n

Bu sorgu veritabanındaki tüm düğümleri getirir ve graf olarak gösterir.

Browser arayüzü, özellikle bilgi grafi sistemlerinde ilişkilerin görsel olarak anlaşılmasını sağlar. Bu, klasik SQL tabanlı sistemlere göre büyük bir farktır çünkü burada veriler tablo değil, ağ olarak görülür.

Düğüm ve İlişki Oluşturma

Neo4j'in temel veri modeli iki ana bileşenden oluşur: düğümler ve ilişkiler.

Bir düğüm oluşturmak için Cypher dili kullanılır:

CREATE (a:Person {name: "Ali", age: 35})

Bu örnekte "Person" etiketine sahip bir düğüm oluşturulmuştur. Düğümün içinde özellikler (properties) bulunur. Bu özellikler, varlıkların niteliklerini temsil eder.

İki düğüm arasındaki ilişki ise şu şekilde oluşturulur:

MATCH (a:Person {name: "Ali"}), (b:Person {name: "Ayşe"})
CREATE (a)-[:FRIENDS_WITH]->(b)

Bu sorgu, Ali ile Ayşe arasında bir "FRIENDS_WITH" ilişkisi kurar. Neo4j'de ilişkiler yönlüdür ve bu yön, bilgi akışını temsil eder.

[13] Sistemde ilk kez bir veri tabanı açtığınızda, yapay zekâ modellerinizi, yazdığınız kodları veya yapacağınız veri analizlerini denemeniz için sistem size otomatik olarak yapay ve izole bir oyun alanı verir. Böylece gerçek verilere zarar verme korkusu yaşamadan, uydurma verilerle sistemin nasıl çalıştığını güvenle deneyimleyebilir, sisteminizi 'ısındırabilirsiniz

Bu model, gerçek dünyadaki ilişkileri çok doğal bir şekilde temsil eder. Örneğin:

- Bir kişi bir şirkette çalışır
- Bir makale başka bir makaleye atıf yapar
- Bir ürün bir kategoriye aittir

Bu tür ilişkiler graf yapısında doğrudan temsil edilir.

Veri İçe Aktarma

Neo4j'in en önemli özelliklerinden biri dış veri kaynaklarından kolayca veri içe aktarabilmesidir. Özellikle CSV dosyaları üzerinden graf oluşturmak oldukça yaygındır.

Örneğin bir CSV dosyasında şu bilgiler olsun:

```
name,friend
Ali,Ayşe
Ayşe,Mehmet
Mehmet,Zeynep
```

Bu veriyi Neo4j'e aktarmak için Cypher kullanılabilir:

```
LOAD CSV WITH HEADERS FROM 'file:///friends.csv' AS row
MERGE (a:Person {name: row.name})
MERGE (b:Person {name: row.friend})
MERGE (a)-[:KNOWS]->(b)
```

Bu işlem, CSV dosyasını okuyarak otomatik olarak düğümler ve ilişkiler oluşturur.

Neo4j ayrıca büyük veri setleri için batch import araçları da sunar. Bu araçlar milyonlarca satırlık veriyi hızlı bir şekilde graf yapısına dönüştürebilir.

Graf Yapısının Görsel Gücü

Neo4j'in en önemli avantajlarından biri, veriyi yalnızca saklaması değil, aynı zamanda görsel olarak sunabilmesidir. Browser arayüzünde düğümler ve ilişkiler bir ağ şeklinde görünür. Bu, özellikle karmaşık veri setlerinde sezgisel analiz yapmayı mümkün kılar.

Örneğin bir sosyal ağ grafiğinde merkezde en çok bağlantıya sahip kişiler hemen fark edilir. Benzer şekilde bir bilgi grafında en çok atıf alan makaleler veya en kritik kavramlar görsel olarak ortaya çıkar.

Sonuç

Neo4j kurulumu ve temel kullanımı, graf veri tabanı dünyasına giriş için en önemli adımlardan biridir. Desktop kurulumu ile başlayan süreç, Browser arayüzü üzerinden sorgular yazma, düğüm ve ilişki oluşturma ve veri içe aktarma işlemleriyle devam eder.

Bu yapı, klasik veri tabanı düşüncesinden farklı olarak ilişkileri merkeze alır. Böylece veri yalnızca saklanmaz, aynı zamanda anlamlı bir ağ yapısı içinde organize edilir. Neo4j, bilgi grafı yaklaşımını pratikte uygulamak için güçlü ve sezgisel bir ortam sunar.

9.4 İlk Cypher Sorguları

Neo4j ile çalışmaya başlandığında en kritik adım, graf veri tabanının sorgu dili olan **Cypher** ile tanışmaktır. Cypher, ilişkisel veri tabanlarındaki SQL'e benzer bir rol oynar; ancak temel farkı, tablo mantığı yerine graf mantığını kullanmasıdır. Yani satır ve sütunlarla değil, düğümler ve ilişkiler üzerinden düşünürsünüz. Bu bölümde Cypher'ın temel yapı taşları olan MATCH, WHERE, traversal (gezinme), path (yol) sorguları ve aggregation (toplulaştırma) işlemleri ele alınacaktır.

MATCH Mantığı

Cypher dilinin kalbi **MATCH** ifadesidir. MATCH, graf içinde desen (pattern) aramak için kullanılır. Bu desenler, düğümler ve ilişkilerden oluşur.

Basit bir örnek:

```
MATCH (n)
RETURN n
```

Bu sorgu, grafteki tüm düğümleri getirir. Ancak Cypher'ın asıl gücü, ilişkileri de içeren desenleri tanımlayabilmesidir.

Örneğin:

```
MATCH (a:Person)-[:FRIENDS_WITH]->(b:Person)
RETURN a, b
```

Bu sorgu, birbirine "FRIENDS_WITH" ilişkisiyle bağlı kişileri getirir.

MATCH mantığı, aslında doğal dilde düşünmeye oldukça yakındır: "Ali'nin arkadaşlarını bul" dediğinizde Cypher bunu doğrudan graf desenine çevirir.

WHERE Filtreleri

MATCH tek başına güçlüdür, ancak çoğu zaman filtreleme gerekir. Bunun için **WHERE** kullanılır.

Örneğin:

```
MATCH (p:Person)
WHERE p.age > 30
RETURN p
```

Bu sorgu, yaşı 30'dan büyük olan kişileri getirir.

WHERE ifadesi, graf üzerinde seçim yapmayı sağlar. Bu seçimler yalnızca düğüm özelliklerine değil, ilişkilere de uygulanabilir.

Örneğin:

```
MATCH (a:Person)-[:FRIENDS_WITH]->(b:Person)
WHERE a.name = "Ali"
RETURN b
```

Bu sorgu, Ali'nin arkadaşlarını getirir.

WHERE filtresi, graf içinde anlamlı alt kümeler oluşturmanın temel aracıdır.

İlişki Gezinme

Graf veri tabanlarının en güçlü özelliklerinden biri **traversal**, yani ağ üzerinde gezinme yeteneğidir. Traversal, bir düğümden başlayarak bağlantılı diğer düğümlere doğru ilerlemeyi ifade eder.

Örneğin:

```
MATCH (a:Person {name: "Ali"})-[:FRIENDS_WITH]->(b)
RETURN b
```

Bu sorgu, Ali'nin doğrudan arkadaşlarını bulur.

Ancak traversal yalnızca bir adımla sınırlı değildir. Daha derin bağlantılar da incelenebilir:

```
MATCH (a:Person {name: "Ali"})-[:FRIENDS_WITH*1..3]->(b)
RETURN b
```

Bu sorgu, Ali'nin 1 ila 3 adım uzaklıktaki bağlantılarını bulur.

Bu özellik, sosyal ağ analizinde veya bilgi grafı sistemlerinde çok önemlidir. Çünkü bilgi çoğu zaman doğrudan değil, dolaylı bağlantılar üzerinden yayılır.

Path Sorguları (Yol Sorguları)

Path sorguları, iki düğüm arasındaki tüm bağlantı yollarını bulmak için kullanılır. Bu, özellikle "en kısa yol" veya "alternatif bağlantı yolları" gibi analizlerde önemlidir.

Örneğin:

```
MATCH path = (a:Person {name:"Ali"})-[:FRIENDS_WITH*]->(b:Person {name:"Zeynep"})
RETURN path
```

Bu sorgu, Ali ile Zeynep arasındaki tüm bağlantı yollarını gösterir.

Path yapıları, graf içinde bilginin nasıl aktığını anlamak için kritik öneme sahiptir. Örneğin bir akademik ağda bir araştırmacının başka bir araştırmacıya nasıl etki ettiğini görmek mümkündür.

Aggregation (Toplulaştırma) İşlemleri

Cypher yalnızca veri getirmek için değil, aynı zamanda veri analizi yapmak için de kullanılır. Bunun için **aggregation** fonksiyonları vardır.

Örneğin:

```
MATCH (p:Person)-[:FRIENDS_WITH]->(f)
RETURN p.name, count(f) AS friend_count
```

Bu sorgu, her kişinin kaç arkadaşı olduğunu hesaplar.

Benzer şekilde:

- count() → sayı hesaplar
- avg() → ortalama alır
- max() / min() → maksimum ve minimum değerleri bulur
- collect() → sonuçları liste haline getirir

Örneğin:

```
MATCH (p:Person)
RETURN avg(p.age)
```

Bu sorgu, ağdaki kişilerin ortalama yaşını hesaplar.

Aggregation işlemleri, graf veriyi analitik hale getirir ve büyük veri analizlerinde kritik rol oynar.

Gerçek Veri Örnekleri

Cypher'ın gerçek gücü, gerçek veri üzerinde çalışıldığında ortaya çıkar. Örneğin bir sosyal ağ verisi düşünelim:

- Kullanıcılar
- Arkadaşlık ilişkileri
- Paylaşımlar

Bu veriyi analiz etmek için Cypher kullanılır.

Örneğin:

```
MATCH (u:User)-[:POSTED]->(p:Post)
WHERE p.likes > 100
RETURN u.name, p.title
```

Bu sorgu, 100'den fazla beğeni alan gönderileri ve yazarlarını bulur.

Başka bir örnek:

```
MATCH (u:User)-[:FOLLOWS]->(f:User)
RETURN u.name, count(f) AS following_count
ORDER BY following_count DESC
```

Bu sorgu, en çok kişiyi takip eden kullanıcıları sıralar.

Bu tür analizler, sosyal medya platformlarında, öneri sistemlerinde ve bilgi grafı uygulamalarında doğrudan kullanılır.

Sonuç

Cypher sorgu dili, graf veri tabanlarını yalnızca veri depolama aracı olmaktan çıkarıp güçlü bir analiz platformuna dönüştürür. MATCH ile desen tanımlanır, WHERE ile filtreleme yapılır, traversal ile ağ üzerinde gezinilir, path sorguları ile bağlantı yolları keşfedilir ve aggregation ile sayısal analiz gerçekleştirilir.

Bu yapı, klasik SQL mantığından farklı olarak ilişkileri merkeze alır. Böylece veri sadece saklanmaz; aynı zamanda bir ağ olarak sorgulanır, analiz edilir ve anlaşılır hale gelir. Cypher, bilgi grafı düşüncesinin pratik programlama dilidir.

9.5 Kùzu Kurulumu ve Python API

Kùzu, son yıllarda ortaya çıkan yeni nesil graf veritabanı yaklaşımlarının en ilginç örneklerinden biridir. Neo4j gibi büyük ve sunucu tabanlı sistemlerin yanında, daha hafif, gömülü ve yerel çalışabilen bir graf veritabanı ihtiyacına cevap vermek için geliştirilmiştir. Özellikle veri bilimciler, araştırmacılar ve tek makine üzerinde çalışan yapay zekâ sistemleri için tasarlanmış olması, onu “kişisel bilgi grafı laboratuvarı” fikrine oldukça uygun hale getirir.

Kùzu Nedir?

Kùzu, graf veri yapısını disk üzerinde verimli şekilde saklayan ve SQL-benzeri bir sorgu diliyle çalışan gömülü bir graf veritabanıdır. Temel amacı, büyük ölçekli dağıtık sistemlere ihtiyaç duymadan, tek bir makine üzerinde yüksek performanslı graf sorguları gerçekleştirmektir.

> Neo4j gibi sistemler genellikle bir sunucu mimarisi gerektirirken, Kùzu doğrudan Python uygulamasının içine gömülebilir. Bu özellik, onu özellikle araştırma ortamları ve prototip geliştirme süreçleri için çok cazip hale getirir.

Kùzu’nun tasarım felsefesi şu üç temel üzerine kuruludur:

- Basitlik
- Yerellik (local-first yaklaşım)
- Yüksek performanslı graf sorgulama

Gömülü Graf Veritabanı Yaklaşımı

Gömülü graf veritabanı (embedded graf database), ayrı bir sunucu çalıştırmadan uygulamanın içinde çalışan veritabanı modelidir. Bu yaklaşımın en önemli avantajı, veri ile uygulama arasındaki gecikmeyi ortadan kaldırmasıdır.

Kùzu bu noktada DuckDB gibi gömülü analitik veritabanlarına benzer bir çizgide yer alır. Nasıl DuckDB SQL analitiğini yerel hale getiriyorsa, Kùzu da graf analitiğini yerel hale getirir.

Bu yaklaşımın sonuçları şunlardır:

- Sunucu kurulumuna gerek kalmaz
- Veri doğrudan dosya sistemi üzerinde tutulur
- Python uygulamasıyla doğrudan entegre çalışır
- Hafif ve hızlı prototipleme yapılabilir

Python Üzerinden Bağlantı Kurma

Kùzu'nun en güçlü yönlerinden biri Python API desteğidir. Kurulum genellikle pip üzerinden yapılır:

```
pip install kuzu
```

Ardından Python içinde bir veritabanı oluşturulur:

```
import kuzu

db = kuzu.Database("mydb")
conn = kuzu.Connection(db)
```

Bu yapı ile artık graf veritabanı Python uygulamasının bir parçası haline gelir. Bağlantı kurulduktan sonra düğümler, ilişkiler ve sorgular doğrudan Python üzerinden yönetilebilir.

CSV Verilerinin Yüklenmesi

Kùzu'nun en pratik özelliklerinden biri CSV dosyalarını doğrudan graf yapıya dönüştürebilmesidir. Örneğin bir "kişiler" veri seti düşünelim:

```
name,age
Ali,35
Ayşe,29
Mehmet,40
```

Bu veri Kùzu'ya şu şekilde yüklenebilir:

```
conn.execute("CREATE NODE TABLE Person(name STRING, age INT, PRIMARY KEY(name))")
```

```
conn.execute("""
COPY Person FROM 'people.csv'
""")
```

Benzer şekilde ilişkiler de yüklenebilir:

```
conn.execute("CREATE REL TABLE FRIENDS(FROM Person TO Person)")
```

Bu yapı sayesinde CSV dosyaları doğrudan graf veri modeline dönüşür. Özellikle veri temizleme ve hazırlık süreçlerinde bu büyük bir hız kazandırır.

İlk Graf Sorguları

Kùzu, SQL'e benzeyen ama graf odaklı bir sorgu dili kullanır. Örneğin:

```
conn.execute("MATCH (p:Person) RETURN p")
```

Bu sorgu tüm kişileri getirir.

İlişki sorgusu:

```
conn.execute("""
MATCH (a:Person)-[:FRIENDS]->(b:Person)
RETURN a.name, b.name
""")
```

Bu sorgu, arkadaşlık ilişkilerini gösterir.

Daha gelişmiş bir örnek:

```
conn.execute("""
MATCH (a:Person)-[:FRIENDS*1..2]->(b:Person)
RETURN a.name, b.name
""")
```

Bu, iki adım uzaklığa kadar tüm bağlantıları keşfeder.

Performans Avantajları

Kùzu'nun en önemli iddialarından biri performanstır. Özellikle yerel disk üzerinde çalıştığı için veri erişimi oldukça hızlıdır. Ayrıca sütun tabanlı depolama ve optimize edilmiş indeks yapıları sayesinde büyük veri setlerinde bile etkili sonuçlar verir.

Neo4j gibi sistemlerde ağ gecikmesi ve sunucu yükü önemli bir faktörken, Kùzu'da bu tür problemler yoktur. Çünkü tüm işlem yerel makinede gerçekleşir.

Performans avantajları şu şekilde özetlenebilir:

- Düşük gecikme (low latency)
- Bellek dostu çalışma

- Disk tabanlı verimli depolama
- Hızlı prototipleme

Local Analytics Yaklaşımı

Kùzu'nun en önemli felsefelerinden biri "local analytics" yani yerel veri analitiğidir. Bu yaklaşım, verinin buluta gönderilmeden doğrudan kullanıcı makinesinde analiz edilmesini hedefler.

Bu özellikle üç alanda kritik önem taşır:

1. **Gizlilik:** Veri dışarı çıkmaz
2. **Hız:** Ağ gecikmesi yoktur
3. **Kontrol:** Tüm süreç kullanıcıdadır

Bu yaklaşım, özellikle akademik araştırmalar, kişisel bilgi grafı sistemleri ve küçük ölçekli yapay zekâ uygulamaları için idealdir.

Kùzu'nun Bilgi Grafı Ekosistemindeki Yeri

Kùzu, Neo4j gibi kurumsal sistemlerin yerine geçmekten ziyade, onları tamamlayan bir araç olarak görülmelidir. Neo4j büyük ölçekli üretim sistemleri için uygunken, Kùzu daha çok araştırma, prototipleme ve kişisel analiz ortamları için tasarlanmıştır.

Özellikle şu senaryolarda öne çıkar:

- Tek makinede çalışan LLM tabanlı sistemler
- Akademik veri analizi
- PDF ve CSV tabanlı bilgi grafı oluşturma
- Yerel RAG (Retrieval-Augmented Generation) sistemleri

Sonuç

Kùzu, graf veri tabanı dünyasında "hafif ama güçlü" yaklaşımın temsilcisidir. Python ile kolay entegrasyonu, CSV verilerini doğrudan graf yapıya dönüştürebilmesi ve yerel analiz yetenekleri sayesinde modern veri bilimi ve yapay zekâ uygulamaları için oldukça önemli bir araç haline gelmiştir.

Bu bölümde görüldüğü gibi Kùzu, yalnızca bir veritabanı değil; aynı zamanda bireysel ölçekte bir bilgi grafı laboratuvarı kurmanın temel bileşenidir. Neo4j'in kurumsal gücüne karşılık, Kùzu kişisel ölçekli zekâ sistemlerinin altyapısını temsil eder.

9.6 PDF ve CSV Verilerinden Bilgi Grafı Oluşturma

Bilgi grafı sistemlerinin en kritik aşamalarından biri, ham verinin anlamlı bir yapıya dönüştürülmesidir. Gerçek dünyadaki verilerin büyük bir kısmı düzenli tablolar halinde değil,

PDF dosyaları, raporlar, makaleler, web sayfaları ve CSV gibi yarı-yapılandırılmış formatlarda bulunur. Bu nedenle bilgi grafı oluşturma süreci çoğu zaman "temiz veriyle başlamak" değil, "dağınık veriden anlam üretmek" problemidir.

Bu bölümde PDF ve CSV gibi iki farklı veri kaynağından nasıl otomatik olarak bilgi grafı üretilebileceği ele alınacaktır. Süreç; yapısız veri problemi, metin çıkarımı, varlık tespiti, ilişki çıkarımı, üstveri üretimi ve otomatik graf oluşturma hattı (pipeline) gibi aşamalardan oluşur.

Yapılandırılmamış Veri Problemi

Yapılandırılmamış veri (unstructured data), önceden tanımlanmış bir şemaya sahip olmayan verilerdir. PDF raporları, akademik makaleler, e-postalar ve web içerikleri bu kategoriye girer. Bu veriler insan tarafından kolayca okunabilirken, makine için doğrudan anlamlı değildir.

Bilgi grafı oluşturmanın temel zorluğu burada başlar: makinenin metin içindeki "kim", "ne", "nerede", "ne yaptı" gibi ilişkileri anlayabilmesi gerekir. Örneğin bir PDF raporunda şu cümle yer alabilir:

"Ali Yılmaz, 2023 yılında Boğaziçi Üniversitesi'nde yapay zekâ üzerine bir proje yürütmüştür."

İnsan bu cümledeki varlıkları ve ilişkiyi hemen anlar. Ancak makine için bu yapı çözülmelidir:

- Ali Yılmaz → kişi
- Boğaziçi Üniversitesi → kurum
- Yapay zekâ projesi → proje
- 2023 → zaman

Bu dönüşüm süreci bilgi grafı oluşturmanın temelini oluşturur.

PDF Parsing Araçları

PDF dosyaları genellikle metin, tablo ve görsel bileşenlerden oluşur. Bu yapıyı makine tarafından okunabilir hale getirmek için "PDF parsing" yani PDF ayrıştırma işlemi yapılır.

Python ekosisteminde yaygın kullanılan bazı araçlar şunlardır:

- PyMuPDF (fitz)
- pdfplumber
- Apache Tika
- unstructured.io

Bu araçlar PDF içindeki metni çıkarır ve düz metin haline getirir. Ancak burada önemli bir sorun vardır: PDF'ler çoğu zaman yapısal olarak tutarlı değildir. Başlıklar, dipnotlar ve sütunlu metinler karışabilir.

Bu nedenle PDF parsing yalnızca metni almak değil, aynı zamanda yapıyı yeniden kurmak anlamına gelir. Örneğin bir akademik makalede “yazar”, “özet”, “giriş” gibi bölümlerin doğru ayrıştırılması gerekir.

CSV’den Varlık Çıkarımı

CSV (Comma Separated Values), yapısal veri formatıdır ve bilgi grafı oluşturmak için en uygun kaynaklardan biridir. Ancak CSV dosyaları genellikle yalnızca ham tablolar içerir; anlam ilişkileri açık değildir.

Örneğin:

name,university,year
Ali Yılmaz,Boğaziçi Üniversitesi,2023

Bu veri doğrudan şu varlıkları içerir:

- Ali Yılmaz (kişi)
- Boğaziçi Üniversitesi (kurum)
- 2023 (zaman)

Varlık çıkarımı, bu ham veriden anlamlı düğümler oluşturma sürecidir. CSV dosyaları bu açıdan “yarı yapılandırılmış bilgi grafı hammaddesi” olarak görülebilir.

İlişki Çıkarma Mantığı

İlişki çıkarımı (relation extraction), bilgi grafı sistemlerinin en kritik adımlarından biridir. Çünkü yalnızca varlıkları bilmek yeterli değildir; bu varlıklar arasındaki bağların da doğru şekilde modellenmesi gerekir.

Örneğin:

“Ali Yılmaz Boğaziçi Üniversitesi’nde çalışmaktadır.”

Bu cümleden şu ilişki çıkarılır:

- (Ali Yılmaz) —[ÇALIŞIR]→ (Boğaziçi Üniversitesi)

Relation extraction iki temel yöntemle yapılabilir:

1. **Kural tabanlı yöntemler:** Dil kalıplarına dayanır
2. **Makine öğrenmesi / LLM tabanlı yöntemler:** Bağlamdan ilişki çıkarır

Modern sistemlerde büyük dil modelleri (LLM) bu süreci büyük ölçüde otomatikleştirmiştir. Model, cümleyi analiz ederek hem varlıkları hem de ilişkileri çıkarabilir.

Metadata Üretimi

Metadata (üstveri), veriyi açıklayan veridir. Bilgi grafı oluştururken yalnızca düğümler ve ilişkiler değil, bu yapıların özellikleri de önemlidir.

Örneğin bir makale için metadata şu bilgileri içerebilir:

- Yazar
- Yayın yılı
- Anahtar kelimeler
- Kaynak
- Atıf sayısı

Bu bilgiler graf yapısına eklendiğinde sistem çok daha zengin hale gelir. Örneğin:

- (Makale) —[YAZAR]→ (Ali Yılmaz)
- (Makale) —[YAYINLANDI]→ (2023)
- (Makale) —[KONU]→ (Yapay Zekâ)

Metadata üretimi, graf sistemin sadece bağlantı değil, anlam katmanı kazanmasını sağlar.

Otomatik Graf Oluşturma Hatı

Tüm bu süreçler birleştiğinde "otomatik bilgi grafı oluşturma hattı" (pipeline) ortaya çıkar. Bu pipeline genellikle şu adımlardan oluşur:

1. **Veri Toplama**
    - PDF, CSV, web kaynakları
2. **Metin Çıkarma**
    - PDF parsing
    - CSV okuma
3. **Varlık Tanıma**
    - Kişi, kurum, yer, tarih tespiti
4. **İlişki Çıkarma**
    - Varlıklar arasındaki bağların belirlenmesi
5. **Metadata Oluşturma**
    - Ek açıklayıcı bilgiler
6. **Graf Yapısına Dönüştürme**
    - Node ve edge oluşturma

7. **Veritabanına Yükleme**
    - Neo4j, Kùzu veya başka bir graf sistemi

Bu pipeline'ın en önemli özelliği otomatikleşebilir olmasıdır. Özellikle büyük dil modelleri sayesinde bu süreç büyük ölçüde insan müdahalesi olmadan gerçekleştirilebilir.

Örnek Senaryo: Akademik PDF'lerden Bilgi Grafı

Bir üniversite kütüphanesinde binlerce PDF makale olduğunu düşünelim. Bu makalelerden otomatik bir bilgi grafı oluşturmak mümkündür:

- Yazarlar düğüm olur
- Makaleler düğüm olur
- Atıflar ilişkiler olur
- Konular kategoriler olur

Örneğin:

- (Ali Yılmaz) —[YAZDI]→ (Makale A)
- (Makale A) —[ATIF YAPTI]→ (Makale B)

Bu yapı sayesinde akademik ağın tamamı görselleştirilebilir ve analiz edilebilir.

Sonuç

PDF ve CSV gibi veri kaynaklarından bilgi grafı oluşturmak, modern veri biliminin en önemli dönüşüm süreçlerinden biridir. Bu süreç, ham ve dağınık veriyi anlamlı bir ağ yapısına dönüştürerek makineye "anlama yeteneği" kazandırır.

Yapısız veri problemi, PDF parsing, varlık ve ilişki çıkarımı, metadata üretimi ve otomatik pipeline tasarımı bir araya geldiğinde, klasik veri işleme yaklaşımından çok daha ileri bir sistem ortaya çıkar: kendi kendine büyüyebilen ve öğrenebilen bir bilgi grafı.

Bu nedenle bu bölüm, yalnızca teknik bir anlatım değil, aynı zamanda verinin anlam haline dönüşme sürecinin pratik karşılığıdır.

9.7 Wikipedia'dan Bilgi Grafı Çıkarmak

Wikipedia, yalnızca bir ansiklopedi değil, aynı zamanda dünyanın en büyük açık bilgi ağlarından biridir. Her sayfa bir kavramı, kişiyi, kurumu veya olayı temsil eder; bu sayfalar arasındaki bağlantılar ise doğal bir bilgi grafı yapısı oluşturur. Bu nedenle Wikipedia, bilgi grafı oluşturma çalışmalarında hem veri kaynağı hem de doğal bir ontoloji örneği olarak kabul edilir.

Bu bölümde Wikipedia'dan otomatik olarak nasıl bilgi grafı çıkarılacağı ele alınacaktır. Süreç; API kullanımı, sayfa bağlantılarının analiz edilmesi, kategori ağlarının çıkarılması, varlıklar arası ilişkilerin modellenmesi ve görselleştirme adımlarını kapsar.

Wikipedia API Kullanımı

Wikipedia, geliştiriciler için güçlü bir API sunar. Bu API sayesinde sayfa içerikleri, bağlantılar, kategoriler ve özet bilgiler programatik olarak çekilebilir.

Python ekosisteminde en yaygın kullanılan araçlardan biri wikipedia ve wikipedia-api kütüphaneleridir. Örneğin bir sayfanın içeriğini çekmek oldukça basittir:

```
import wikipedia

page = wikipedia.page("Yapay zeka")
print(page.content)
```

Bu komut, “Yapay zeka” sayfasının tüm metin içeriğini getirir.

Daha gelişmiş API kullanımıyla sayfa başlıkları, bağlantılar ve özetler ayrı ayrı çekilebilir. Bu yapı, bilgi grafı oluşturma sürecinin ilk adımını oluşturur: veri toplama.

Sayfa Bağlantıları (Hyperlink Graf)

Wikipedia’nın en güçlü yönlerinden biri, sayfalar arasındaki hiperlink yapısıdır. Her sayfa, diğer sayfalara bağlantılar içerir ve bu bağlantılar doğal bir graf yapısı oluşturur.

Örneğin “Yapay zeka” sayfasında şu bağlantılar olabilir:

- Makine öğrenmesi
- Derin öğrenme
- Veri bilimi
- Algoritmalar

Bu durumda şu graf ilişkisi oluşur:

- (Yapay Zeka) → (Makine Öğrenmesi)
- (Yapay Zeka) → (Derin Öğrenme)

Bu yapı, Wikipedia’yı devasa bir yönlü graf haline getirir. Her bağlantı bir bağlantı (edge), her sayfa bir düğüm (node) olarak temsil edilebilir.

Bu bağlantılar çıkarıldığında, Wikipedia aslında otomatik olarak oluşmuş bir bilgi grafı haline gelir.

Kategori Ağları

Wikipedia sayfaları yalnızca bağlantılarla değil, aynı zamanda kategorilerle de organize edilir. Her sayfa bir veya daha fazla kategoriye aittir.

Örneğin:

- “Yapay zeka” → “Bilgisayar bilimi”
- “Albert Einstein” → “Fizikçiler”
- “İkinci Dünya Savaşı” → “Tarihsel olaylar”

Bu kategoriler ikinci bir graf katmanı oluşturur. Böylece iki farklı ağ yapısı ortaya çıkar:

1. **Sayfa bağlantı ağı (link graf)**
2. **Kategori ağı (taxonomy graf)**

Kategori ağı, daha üst düzey bir semantik yapı sağlar. Bu yapı, ontolojiye oldukça yakındır çünkü kavramların hiyerarşik ilişkilerini gösterir.

İnsanlar–Kurumlar–Olaylar İlişkileri

Wikipedia’dan çıkarılabilecek en değerli bilgi türlerinden biri varlıklar arasındaki ilişkileridir. Özellikle üç ana kategori öne çıkar:

- İnsanlar (People)
- Kurumlar (Organizations)
- Olaylar (Events)

Örneğin:

- Albert Einstein → Princeton Üniversitesi
- Albert Einstein → Görelilik Teorisi
- İkinci Dünya Savaşı → Birleşmiş Milletler’in kurulması

Bu ilişkiler şu şekilde modellenebilir:

- (Einstein) —[ÇALIŞTI]→ (Princeton Üniversitesi)
- (Einstein) —[GELİŞTİRDİ]→ (Görelilik Teorisi)

Bu yapı, bilgi grafının temel omurgasını oluşturur. Çünkü gerçek dünya bilgisi büyük ölçüde bu üçlü ilişkiler üzerine kuruludur.

Mini Semantic Web Örneği

Wikipedia’dan çıkarılan graf aslında küçük ölçekli bir “anlamsal web” (semantic web) örneğidir. Anlamsal web’in amacı, interneti yalnızca belgelerden oluşan bir yapı olmaktan çıkarıp, anlamlı ilişkiler ağı haline getirmektir.

Wikipedia grafı bu açıdan doğal bir örnek sunar:

- Sayfalar → varlıklar (entities)
- Bağlantılar → ilişkiler (relations)

- Kategoriler → ontolojik sınıflar (classes)

Bu yapı RDF (Resource Description Framework) modeline oldukça yakındır:

- Subject → Wikipedia sayfası
- Predicate → bağlantı türü
- Object → hedef sayfa

Örneğin:

- (Yapay Zeka) — “alt alanıdır” → (Bilgisayar Bilimi)

Bu yapı, semantik web’in pratik karşılığıdır.

Görselleştirme Çalışması

Wikipedia’dan çıkarılan bilgi grafı görselleştirildiğinde, oldukça karmaşık ama anlamlı bir ağ yapısı ortaya çıkar. Bu görselleştirme için Python’da NetworkX veya Neo4j gibi araçlar kullanılabilir.

Basit bir örnek:

```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graf()

G.add_edge("Yapay Zeka", "Makine Öğrenmesi")
G.add_edge("Yapay Zeka", "Veri Bilimi")
G.add_edge("Makine Öğrenmesi", "Derin Öğrenme")

nx.draw(G, with_labels=True)
plt.show()
```

Daha büyük ölçekli bir sistemde binlerce düğüm ve milyonlarca bağlantı görselleştirilebilir. Bu durumda kümeler, merkez düğümler ve alt alanlar net bir şekilde ortaya çıkar.

Görselleştirme, yalnızca estetik bir araç değil, aynı zamanda analitik bir yöntemdir. Çünkü insan beyni ilişkisel yapıları tabloya göre çok daha hızlı algılar.

Sonuç

Wikipedia’dan bilgi grafı çıkarmak, modern bilgi mühendisliğinin en güçlü tekniklerinden biridir. API kullanımı sayesinde veri çekilir, sayfa bağlantılarıyla graf yapısı oluşturulur, kategorilerle semantik katman eklenir ve varlık ilişkileri ile gerçek dünya bilgisi modellenir.

Bu süreç sonunda Wikipedia, yalnızca bir ansiklopedi olmaktan çıkar; yaşayan, büyüyen ve analiz edilebilen bir bilgi ağına dönüşür. Bu ağ, hem yapay zekâ sistemleri için eğitim verisi hem de bilgi grafı sistemleri için temel bir yapı taşıdır.

9.8 ChatGPT + Bilgi Grafı Uygulaması

Büyük dil modelleri (LLM'ler), özellikle ChatGPT gibi sistemler, metin üretme konusunda son derece güçlüdür; ancak bu güç, her zaman "bilgiye dayanma" anlamına gelmez. Bu modeller, büyük ölçüde olasılıksal dil örüntülerini öğrenerek çalışır. Bu nedenle doğru görünen ama gerçekte yanlış olan ifadeler üretebilirler. İşte bu noktada bilgi grafı yaklaşımı devreye girer ve LLM'lerin eksik yönlerini tamamlayan yapısal bir bellek katmanı sağlar.

Bu bölümde ChatGPT benzeri modeller ile bilgi grafı sistemlerinin nasıl birlikte çalıştığı, neden bu entegrasyonun kritik olduğu ve pratik uygulamalarda nasıl kullanıldığı ele alınacaktır.

LLM'ler Neden Graf Kullanır?

Büyük dil modelleri temel olarak metin üzerinde eğitilir ve dünyayı "kelime dizileri" üzerinden öğrenir. Ancak gerçek dünya yalnızca metinsel dizilerden oluşmaz; varlıklar ve bu varlıklar arasındaki ilişkilerden oluşan bir ağ yapısı vardır.

Örneğin:

- Einstein fizikçidir
- Einstein görelilik teorisini geliştirmiştir
- Görelilik teorisi fizik ile ilgilidir

Bu bilgiler LLM içinde dağınık şekilde bulunabilir, ancak graf yapısında açık ve sorgulanabilir hale gelir.

Graf kullanımı LLM'lere şu avantajları sağlar:

- Yapısal bellek
- Tutarlı bilgi temsili
- Daha az hata üretimi
- Açıklanabilir cevaplar

Bu nedenle modern yapay zekâ sistemleri giderek daha fazla "hibrit mimari"ye yönelmektedir: LLM + bilgi grafı.

Graf Tabanlı Bellek Yaklaşımı

LLM'lerin en büyük sınırlamalarından biri kalıcı belleklerinin olmamasıdır. Her oturum bağımsızdır ve önceki bilgiler doğrudan hatırlanmaz. Bu problemi çözmek için graf tabanlı bellek sistemleri geliştirilmiştir.

Bu yaklaşımda:

- Varlıklar düğüm olarak saklanır
- İlişkiler bağlantı (olarak temsil edilir
- Modelin ürettiği bilgiler grafa yazılır
- Model gerektiğinde grafı sorgular

Örneğin:

- (Kullanıcı) —[İLGİLENİR]→ (Bilgi Grafı)
- (Kullanıcı) —[OKUDU]→ (Kitap A)

Bu yapı sayesinde ChatGPT benzeri bir sistem kullanıcıyı "hatırlayan" bir asistana dönüşebilir.

Graf tabanlı hafıza, özellikle kişiselleştirilmiş yapay zekâ sistemlerinin temelini oluşturur.

Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG), büyük dil modellerini dış bilgi kaynaklarıyla birleştiren bir yaklaşımdır. Bu modelde LLM, cevap üretmeden önce ilgili bilgileri bir veri kaynağından "geri getirir" (retrieval).

Süreç şu şekilde işler:

1. Kullanıcı soru sorar
2. Sistem bilgi grafı veya veri tabanından ilgili düğümleri bulur
3. Bu bilgiler LLM'e bağlam olarak verilir
4. LLM bu bağlamı kullanarak cevap üretir

Örneğin:

Soru: "Einstein hangi üniversitelerde çalıştı?"

Sistem önce grafı sorgular:

- Einstein → Princeton Üniversitesi
- Einstein → Zürih Üniversitesi

Sonra LLM bu bilgiyi kullanarak doğru ve tutarlı bir cevap üretir.

RAG yaklaşımı, LLM'lerin "bilgiye dayalı üretim" yapmasını sağlar.

Hallucination Azaltma Yöntemleri

Hallucination (uydurma bilgi üretimi), LLM'lerin en önemli problemlerinden biridir. Model, emin olmadığı durumlarda bile akıcı ama yanlış bilgiler üretebilir.

Bilgi grafı entegrasyonu bu problemi önemli ölçüde azaltır.

Bunun birkaç nedeni vardır:

- Model dış kaynaktan doğrulanmış bilgi alır
- Graf yapısı tutarlı ilişkiler sağlar
- Yanlış bağlantılar filtrelenebilir

Örneğin, graf tabanlı sistemde olmayan bir ilişki model tarafından üretildiğinde bu bilgi doğrulanmaz ve cevap dışı bırakılır.

Ek olarak şu yöntemler kullanılır:

- Graf validation (graf doğrulama)
- Confidence scoring (güven skoru)
- Multi-source verification (çoklu kaynak kontrolü)

Bu sayede model daha güvenilir hale gelir.

Chatbot'a Bilgi Grafı Bağlamak

ChatGPT benzeri bir chatbot'a bilgi grafı bağlamak, modern YZ mimarilerinde oldukça yaygın bir yaklaşımdır.

Temel yapı şu şekilde kurulur:

- Kullanıcı arayüzü (chatbot)
- LLM (ChatGPT)
- Bilgi grafı (Neo4j, Kùzu vb.)
- Retrieval katmanı

Sistem şöyle çalışır:

1. Kullanıcı soru sorar
2. Soru semantik olarak analiz edilir
3. Bilgi grafında ilgili düğümler aranır
4. Sonuçlar LLM'e bağlam olarak verilir
5. Model cevap üretir

Bu yapı sayesinde chatbot yalnızca dil üretmez, aynı zamanda "bilgiye dayalı akıl yürütme" yapar.

Örneğin:

Soru: “Yapay zekâ alanında çalışan Türk bilim insanları kimlerdir?”

Graf sorgusu:

- (Bilim İnsanı) —[ÇALIŞMA ALANI]→ (Yapay Zekâ)
- (Bilim İnsanı) —[UYRUK]→ (Türkiye)

Semantik Sorgulama Örnekleri

Bilgi grafı + LLM entegrasyonunun en güçlü yönlerinden biri semantik sorgulama yeteneğidir. Kullanıcı SQL veya Cypher bilmeden doğal dil ile sorgu yapabilir.

Örneğin:

Soru: “Einstein’ın çalıştığı kurumlar nelerdir?”

Sistem bunu şu graf sorgusuna çevirir:

```
MATCH (e:Person {name:"Einstein"})-[:WORKED_AT]->(c)
RETURN c
```

Başka bir örnek:

Soru: “Makine öğrenmesi ile ilgili en çok bağlantıya sahip kavramlar hangileridir?”

Sorgu:

```
MATCH (m:Concept {name:"Machine Learning"})--(n)
RETURN n, count(*) AS degree
ORDER BY degree DESC
```

Bu tür sorgular, doğal dil ile graf veritabanı arasında köprü kurar.

Sonuç

ChatGPT ile bilgi grafı entegrasyonu, modern yapay zekânın en önemli evrim adımlarından biridir. LLM’ler dil üretiminde güçlüdür, ancak bilgi doğruluğu ve yapısal tutarlılık açısından sınırlıdır. Bilgi grafı ise tam tersine, yüksek doğruluk ve yapısal tutarlılık sunar.

Bu iki yaklaşım birleştirildiğinde ortaya hibrit bir zeka sistemi çıkar:

- Dilsel akıcılık (LLM)
- Yapısal doğruluk (Graf)
- Bellek sürekliliği (Bilgi grafı)
- Doğrulanabilir bilgi (RAG)

Bu mimari, yalnızca daha akıllı chatbotlar değil, aynı zamanda açıklanabilir, güvenilir ve kurumsal düzeyde kullanılabilir yapay zekâ sistemleri üretmenin temel yoludur.

9.9 Akademik Makalelerden Bilgi Ağı Kurmak

Akademik dünya, yüz binlerce dergi, milyonlarca makale ve milyonlarca araştırmacıdan oluşan dev bir bilgi ekosistemidir. Bu ekosistemin yüzeyde görünen yapısı yayınlar, atıflar ve yazar isimlerinden ibarettir. Ancak derin yapıya bakıldığında, aslında bu sistemin tamamen bir bilgi grafı olduğu görülür. Her makale bir düğüm, her yazar bir düğüm, her atıf ise bu düğümler arasında bir ilişki olarak temsil edilebilir. Bu nedenle akademik literatür, doğal olarak graf tabanlı analiz için en uygun veri alanlarından biridir.

Bu bölümde akademik makalelerden nasıl bir bilgi ağı kurulabileceği; DOI sistemi, bibliyometrik veri, atıf ağları, yazar işbirliği ağları, konu kümeleri ve bilim haritaları üzerinden ele alınacaktır.

DOI ve Bibliyometrik Veri

DOI (Digital Object Identifier), akademik bir makalenin dijital dünyadaki kimlik numarasıdır. Her DOI benzersizdir ve bir makaleyi kalıcı olarak temsil eder. Bu özellik, akademik veri grafı oluşturmanın temelini sağlar.

Örneğin:

- DOI: 10.1038/nature12373

Bu DOI üzerinden şu bilgiler çekilebilir:

- Yazarlar
- Yayın yılı
- Dergi
- Atıf sayısı
- Anahtar kelimeler

Bibliyometrik veri ise akademik yayınların nicel analizini ifade eder. Bu veriler şunları içerir:

- Atıf sayıları
- H-indeksi
- Yazar üretkenliği
- Dergi etki faktörleri

Bu bilgiler graf yapısına dönüştürüldüğünde, akademik dünyanın matematiksel bir haritası ortaya çıkar.

Atıf Ağları

Atıf ağları, akademik bilgi grafının en temel bileşenidir. Bir makale başka bir makaleyi referans gösterdiğinde, bu iki makale arasında yönlü bir bağlantı oluşur.

Örneğin:

- Makale A → Makale B (A, B'yi atıf yapar)

Bu yapı zaman içinde devasa bir ağ oluşturur. Bu ağın özellikleri:

- Yönlüdür (directed graf)
- Zamana bağlıdır (temporal structure)
- Ölçekten bağımsızdır (scale-free)

Atıf ağları sayesinde:

- En etkili makaleler belirlenebilir
- Bilimsel alanların gelişimi izlenebilir
- Paradigma değişimleri tespit edilebilir

Örneğin yapay zekâ alanında "transformer" makalesi, kısa sürede yüksek merkeziyete ulaşarak ağın yapısını değiştirmiştir.

Yazar İşbirliği Ağları

Akademik üretim yalnızca makalelerden değil, aynı zamanda insanlar arasındaki işbirliklerinden oluşur. Yazar işbirliği ağları (co-authorship networks), kimlerin birlikte çalıştığını gösterir.

Bu ağlarda:

- Düğümler: Yazarlar
- Bağlantılar: Ortak yazılmış makaleler

Örneğin:

- Ali Yılmaz — Mehmet Demir (ortak makale yazmış)

Bu yapı sayesinde:

- Bilimsel ekipler belirlenebilir
- Disiplinler arası köprüler görülebilir
- Akademik topluluklar analiz edilebilir

Yazar ağları genellikle "küçük dünya ağı" özellikleri gösterir. Yani çoğu araştırmacı birkaç adımda birbirine bağlanabilir.

Konu Kümeleri (Topic Clusters)

Akademik makaleler yalnızca yazarlar ve atıflar üzerinden değil, aynı zamanda konu yapıları üzerinden de analiz edilebilir. Bu süreçte doğal dil işleme (NLP) ve gömme yöntemleri kullanılır.

Örneğin makaleler şu konulara ayrılabilir:

- Makine öğrenmesi
- Veri bilimi
- Biyoinformatik
- Kuantum hesaplama

Bu konular bir graf yapısı içinde kümeler oluşturur. Her küme, belirli bir bilimsel alanı temsil eder.

Bu yapı sayesinde:

- Araştırma alanları otomatik sınıflandırılır
- Disiplinler arası bağlantılar ortaya çıkar
- Yeni araştırma alanları tespit edilir

Örneğin "graf sinir ağları" (GNN), hem graf teorisi hem de makine öğrenmesi kümeleri arasında bir köprü oluşturur.

Bilim Haritaları Üretmek

Bilim haritası, akademik verilerin görsel olarak temsil edilmesidir. Bu haritalar, bilimsel alanların nasıl geliştiğini ve birbirleriyle nasıl ilişkili olduğunu gösterir.

Bilim haritaları genellikle şu verilerden oluşturulur:

- Atıf ağları
- Yazar ağları
- Anahtar kelime ağları
- Dergi ağları

Bu veriler birleştirildiğinde, bilimsel bilginin topolojisi ortaya çıkar.

Örneğin:

- Yapay zekâ → veri bilimi → istatistik → matematik

Bu zincir, alanlar arasındaki bilgi akışını gösterir.

Bilim haritaları özellikle:

- Araştırma stratejisi geliştirme
- Fon dağılımı analizi
- Akademik trend tahmini

gibi alanlarda kullanılır.

Akademik Keşif Sistemleri

Akademik bilgi grafı yalnızca analiz için değil, aynı zamanda keşif için de kullanılır. Bu sistemlere "akademik keşif sistemleri" denir.

Bu sistemler şu soruları cevaplayabilir:

- "Bu konuda en önemli makaleler hangileri?"
- "Bu iki alan arasında bağlantı var mı?"
- "Hangi araştırmacılar benzer konularda çalışıyor?"

Örneğin bir araştırmacı "graf sinir ağları" ile ilgileniyorsa sistem ona şu tür öneriler sunabilir:

- Benzer çalışmalar
- İlgili yazarlar
- Yeni ortaya çıkan makaleler

Bu yapı, Google Scholar'ın ötesinde daha derin bir semantik arama deneyimi sunar.

Sonuç

Akademik makalelerden bilgi ağı kurmak, bilimi yalnızca bir yayın koleksiyonu olarak değil, yaşayan bir ağ sistemi olarak görmeyi sağlar. DOI, atıf ağları, yazar ilişkileri ve konu kümeleri bir araya geldiğinde, bilimsel bilginin görünmeyen mimarisi ortaya çıkar.

Bu yapı sayesinde:

- Bilimsel etki ölçülebilir
- Araştırma alanları haritalanabilir
- Yeni keşifler öngörülebilir

Sonuç olarak akademik dünya, aslında devasa bir bilgi grafıdır. Bu grafı anlamak, bilimin nasıl ilerlediğini anlamak demektir.

9.10 Kurumsal Bilgi Grafı Tasarımı

Kurumsal bilgi grafı , bir organizasyonun sahip olduğu tüm veriyi yalnızca depolayan değil, aynı zamanda bu veriler arasındaki anlamlı ilişkileri ortaya çıkaran bütünleşik bir yapıdır. Geleneksel veri tabanları genellikle departmanlara göre bölünmüş, birbirinden kopuk sistemler halinde çalışır. Finans, insan kaynakları, satış, üretim ve müşteri ilişkileri yönetimi gibi alanlar kendi veri silolarını üretir. Kurumsal bilgi grafı yaklaşımı ise bu siloları ortadan kaldırarak, veriyi tek bir anlam ağı içinde birleştirmeyi hedefler.

Bu bölümde kurumsal bilgi grafı tasarımının temel bileşenleri ele alınacaktır: şirket içi veri kaynakları, ERP/CRM/IoT entegrasyonu, ontoloji tasarımı, veri yönetişimi, semantik katman ve dijital ikiz yaklaşımı.

Şirket İçi Veri Kaynakları

Bir kurumun bilgi grafını oluşturmanın ilk adımı, mevcut veri kaynaklarını tanımlamaktır. Modern şirketlerde veri çok farklı sistemlerde dağınık halde bulunur:

- ERP sistemleri (kurumsal kaynak planlama)
- CRM sistemleri (müşteri ilişkileri yönetimi)
- İnsan kaynakları sistemleri
- Finansal kayıtlar
- E-posta ve belge arşivleri
- IoT sensör verileri

Bu kaynakların ortak problemi, birbirleriyle doğrudan konuşamıyor olmalarıdır. Örneğin CRM sisteminde bir müşteri bilgisi bulunurken, ERP sisteminde aynı müşterinin sipariş verisi yer alır. Ancak bu iki veri genellikle farklı anahtarlarla tutulur.

Bilgi grafı, bu parçalanmış yapıyı birleştirerek müşteriyi, siparişi, ürünü ve etkileşimleri tek bir ağ içinde temsil eder.

ERP, CRM ve IoT Entegrasyonu

Kurumsal bilgi grafının en kritik aşamalarından biri farklı sistemlerin entegrasyonudur.

ERP (Enterprise Resource Planning) sistemleri genellikle üretim, stok ve finans verilerini içerir. CRM (Customer Relationship Management) sistemleri müşteri etkileşimlerine odaklanır. IoT (Internet of Things) sistemleri ise sensörlerden gelen gerçek zamanlı verileri sağlar.

Bu üç veri kaynağı bilgi grafında şu şekilde birleşebilir:

- Müşteri → sipariş verir → ürün
- Ürün → üretim hattında üretilir → fabrika
- Fabrika → IoT sensörü → sıcaklık verisi

Bu entegrasyon sayesinde şirket yalnızca geçmiş veriyi değil, gerçek zamanlı operasyonel durumu da anlayabilir.

Ontoloji Tasarımı

Ontoloji (ontology), bir bilgi grafının kavramsal iskeletidir. Hangi varlıkların bulunduğunu, bu varlıkların hangi özelliklere sahip olduğunu ve birbirleriyle nasıl ilişkili olduğunu tanımlar.

Kurumsal bir ontoloji genellikle şu kavramları içerir:

- Müşteri
- Ürün
- Sipariş
- Tedarikçi
- Çalışan
- Lokasyon
- Süreç

Her kavramın özellikleri ve ilişkileri tanımlanır. Örneğin:

- Müşteri → sipariş verir → Ürün
- Çalışan → çalışır → Departman
- Ürün → üretilir → Fabrika

Ontoloji tasarımı, bilgi grafının tutarlılığını sağlar. Eğer iyi tasarlanmamış bir ontoloji varsa, graf yapı karmaşık ve kullanılamaz hale gelir.

Veri Yönetişimi

Kurumsal bilgi grafı yalnızca teknik bir yapı değildir; aynı zamanda bir yönetim problemidir. Veri yönetişimi, verinin kim tarafından üretildiği, nasıl güncellendiği ve nasıl kullanıldığı ile ilgilidir.

Bilgi grafı sistemlerinde veri yönetişimi şu soruları cevaplar:

- Hangi veri güvenilirdir?
- Hangi kaynak “doğru” kabul edilir?
- Veriyi kim güncelleyebilir?
- Çakışan bilgiler nasıl çözülür?

Örneğin CRM sisteminde bir müşterinin adresi güncellenmiş, ancak ERP sisteminde eski adres kalmış olabilir. Bilgi grafı bu çelişkiyi çözmek zorundadır.

Bu nedenle veri yönetişimi, teknik olduğu kadar organizasyonel bir konudur.

Kurumsal Semantik Katman

Semantik katman, ham veriyi iş anlamına dönüştüren soyutlama katmanıdır. Bu katman olmadan graf yapı yalnızca teknik bir veri ağı olur; ancak semantik katman ile birlikte anlamlı bir iş modeline dönüşür.

Örneğin:

- “Sipariş” yalnızca bir kayıt değildir
- Aynı zamanda bir müşteri davranışıdır
- Ve gelir üretim sürecinin bir parçasıdır

Semantik katman şu avantajları sağlar:

- Kullanıcıların teknik detayları bilmeden veri sorgulaması
- Ortak iş dili oluşturulması
- Farklı sistemlerin aynı anlamda konuşması

Bu katman sayesinde “veri” yerine “anlamlı bilgi” sorgulanır.

Dijital İkiz Yaklaşımı

Kurumsal bilgi grafının en ileri kullanım alanlarından biri dijital ikiz (digital twin) yaklaşımıdır. Dijital ikiz, gerçek dünyadaki bir sistemin sanal ortamda birebir temsilidir.

Bir fabrikanın dijital ikizi şu bileşenlerden oluşabilir:

- Üretim makineleri (düğüm)
- Sensörler (düğüm)
- Üretim süreçleri (ilişki)
- Gerçek zamanlı veriler (IoT stream)

Bu yapı sayesinde:

- Makinenin anlık durumu izlenebilir
- Arızalar önceden tahmin edilebilir
- Üretim optimizasyonu yapılabilir

Bilgi grafı burada yalnızca veri modelleme aracı değil, aynı zamanda simülasyon platformu haline gelir.

Kurumsal Bilgi Grafının İş Değeri

Kurumsal bilgi grafı sistemleri, şirketlere yalnızca teknik avantaj değil, stratejik avantaj da sağlar:

- Daha hızlı karar alma
- Veri silolarının ortadan kalkması
- Müşteri davranışlarının daha iyi anlaşılması
- Operasyonel verimlilik artışı

- Risk analizi ve erken uyarı sistemleri

Örneğin bir bankada:

- Müşteri hesapları
- Kredi geçmişi
- Dolandırıcılık sinyalleri

tek bir graf yapıda birleştiğinde, dolandırıcılık tespiti çok daha hızlı yapılabilir.

Sonuç

Kurumsal bilgi grafı tasarımı, modern işletmelerin veriyle ilişkisinin kökten değişmesini temsil eder. Artık veri yalnızca saklanan bir varlık değil, ilişkiler ağı içinde anlam kazanan bir yapıdır.

ERP, CRM ve IoT gibi sistemlerin entegrasyonu, ontoloji tasarımı, veri yönetişimi ve semantik katman birlikte çalışarak şirketin dijital zekâsını oluşturur. Dijital ikiz yaklaşımı ise bu yapıyı gerçek zamanlı bir simülasyon sistemine dönüştürür.

Sonuç olarak kurumsal bilgi grafı, yalnızca bir teknoloji değil, organizasyonun kendisini yeniden tanımlama biçimidir.

9.11 Mini Proje: LLM Destekli Akademik Arama Motoru

Bu bölümde, önceki bölümlerde anlatılan tüm kavramların küçük ama uçtan uca bir uygulaması olarak "LLM destekli akademik arama motoru" tasarlanacaktır. Amaç, yalnızca anahtar kelimeyle çalışan klasik arama sistemlerinden farklı olarak, anlamı anlayan, ilişkileri takip eden ve hem metin hem de graf yapılarını birlikte kullanan bir akademik bilgi sistemi inşa etmektir. Bu sistem; bilgi grafı, vektör veritabanı ve büyük dil modeli (LLM) bileşenlerini birleştiren hibrit bir mimariye dayanır.

Makale Koleksiyonu Oluşturmak

Projenin ilk adımı, bir akademik makale koleksiyonu oluşturmaktır. Bu koleksiyon küçük bir veri seti ile başlayabilir; örneğin 500–1000 makalelik bir corpus.

Bu makaleler şu kaynaklardan toplanabilir:

- OpenAlex veri seti
- arXiv API
- Semantic Scholar API
- PDF arşivleri

Her makale şu temel bileşenleri içerir:

- Başlık

- Özet (abstract)
- Yazarlar
- Yayın yılı
- Anahtar kelimeler
- Atıf bilgileri

Bu veriler hem metin tabanlı analiz hem de graf modelleme için temel oluşturur. Her makale bir düğüm (node) olarak temsil edilir ve yazarlar, konular ve atıflar bu düğümlerle ilişkilendirilir.

Gömme Üretimi

Bir akademik arama motorunun en kritik bileşenlerinden biri gömme üretimidir. Gömme, metinleri sayısal vektörlere dönüştürerek anlam benzerliğini matematiksel olarak temsil eder.

Örneğin:

- "machine learning" ve "deep learning" kavramları benzer gömme vektörlerine sahiptir
- Bu sayede sistem yalnızca kelime eşleşmesi değil, anlam benzerliği üzerinden arama yapabilir

Gömme üretimi için genellikle şu modeller kullanılır:

- Sentence-BERT
- OpenAI text embedding modelleri
- E5 veya BGE modelleri

Her makale için gömme oluşturulur ve bu vektörler bir vektör veritabanında saklanır.

Semantic Arama Yaklaşımı

Semantik arama (semantic search), klasik anahtar kelime aramasının ötesine geçer. Kullanıcı bir soru sorduğunda sistem kelimeleri değil, anlamı analiz eder.

Örneğin:

Soru: "derin öğrenme ile görüntü tanıma çalışmaları"

Sistem yalnızca "derin öğrenme" ve "görüntü tanıma" kelimelerini aramaz; bu kavramlara semantik olarak yakın makaleleri de getirir.

Bu süreç şu şekilde çalışır:

1. Soru gömmeye dönüştürülür
2. Makale gömmeleri ile karşılaştırılır
3. En yakın vektörler bulunur

4. İlgili makaleler geri getirilir

Bu yaklaşım, özellikle akademik aramalarda büyük bir kalite artışı sağlar.

Local RAG Sistemi

Retrieval-Augmented Generation (RAG), LLM'lerin dış bilgi kaynaklarıyla birlikte çalışmasını sağlayan bir mimaridir. Local RAG ise bu sistemi yerel bir ortamda çalıştırır.

Bu projede RAG süreci şu şekilde işler:

1. Kullanıcı soru sorar
2. Sistem gömme ve/veya graf sorgusu yapar
3. İlgili makaleler seçilir
4. Bu içerik LLM'e bağlam olarak verilir
5. LLM cevap üretir

Örneğin:

Soru: "Graf neural networks nerelerde kullanılır?"

Sistem önce ilgili makaleleri getirir, sonra LLM bu makaleleri kullanarak açıklayıcı bir cevap üretir.

Bu yöntem, uydurma bilgi problemini de ciddi şekilde azaltır.

Graf + Vector Database Hibrit Yapısı

Bu projenin en önemli yönlerinden biri hibrit mimaridir. Burada iki farklı veri yapısı birlikte kullanılır:

1. **Vector database (vektör veritabanı):**
    - Anlamsal benzerlik araması yapar
    - Makaleleri embedding üzerinden bulur
2. **Graf database (graf veritabanı):**
    - Yazarlar, atıflar ve konular arasındaki ilişkileri tutar
    - Yapısal analiz sağlar

Bu iki yapı birlikte kullanıldığında sistem hem "anlam" hem de "ilişki" düzeyinde çalışabilir.

Örneğin:

- Vektör araması: benzer makaleleri bulur

- Graf arama: bu makalelerin kimler tarafından yazıldığını ve nasıl bağlantılı olduğunu gösterir

Bu kombinasyon, akademik arama motorunu çok daha güçlü hale getirir.

Akademik Soru-Cevap Sistemi

Bu sistemin en görünür kısmı, kullanıcıların doğal dilde soru sorabildiği akademik chatbot bileşenidir.

Örneğin:

Soru: “Transformer modelleri hangi alanlarda kullanılıyor?”

Sistem şu adımları uygular:

1. Soru analiz edilir
2. Vektör araması yapılır
3. Graf sorgusu ile ilgili alanlar bulunur
4. LLM bağlamı kullanarak cevap üretir

Sonuç olarak kullanıcıya yalnızca liste değil, açıklamalı ve bağlamsal bir cevap verilir.

Örneğin:

- Doğal dil işleme
- Görüntü işleme
- Biyoinformatik
- Zaman serisi analizi

Sonuçların Görselleştirilmesi

Bu tür bir sistemin en güçlü yönlerinden biri görselleştirme katmanıdır. Akademik bilgi grafı yalnızca metin olarak değil, görsel bir ağ olarak da sunulabilir.

Görselleştirme ile:

- Yazarlar arası işbirliği ağları görülebilir
- Makaleler arasındaki atıf zincirleri analiz edilebilir
- Konu kümeleri (topic clusters) keşfedilebilir

Örneğin:

- Büyük düğümler → yüksek atıf alan makaleler
- Yoğun bölgeler → araştırma alanları

- Köprü düğümler → disiplinler arası çalışmalar

Bu görsel yapı, araştırmacıya sezgisel bir analiz imkânı sunar.

Projenin Genel Mimarisi

Bu mini proje şu bileşenlerden oluşur:

- Veri katmanı: Makaleler, metadata
- Vektör katmanı: Gömme tabanlı arama
- Graf katmanı: Neo4j / Kùzu
- LLM katmanı: ChatGPT benzeri model
- API katmanı: Sorgu yönetimi
- UI katmanı: arayüz ve görselleştirme

Bu yapı birlikte çalışarak modern bir "akademik zekâ sistemi" oluşturur.

Sonuç

LLM destekli akademik arama motoru, bilgi grafı ve yapay zekâ entegrasyonunun en pratik örneklerinden biridir. Bu sistem yalnızca bilgi bulmaz; aynı zamanda bilgiyi bağlam içinde açıklar, ilişkileri gösterir ve kullanıcıya yeni keşif yolları sunar.

Gömme tabanlı semantik arama, graf tabanlı ilişkisel analiz ve LLM tabanlı açıklama üretimi bir araya geldiğinde, klasik arama motorlarının çok ötesinde bir yapı ortaya çıkar.

Bu mini proje, aslında daha büyük bir fikrin küçük bir prototipidir: bilginin yalnızca depolanmadığı, aynı zamanda anlamlı bir ağ içinde sürekli olarak yeniden üretildiği bir dijital zekâ sistemi.

9.12 Tek Kişilik Bilgi Grafı Laboratuvarı

Bilgi grafı teknolojileri uzun süre boyunca büyük kurumların, teknoloji şirketlerinin ve araştırma merkezlerinin alanı olarak görülmüştür. Ancak günümüzde hem açık kaynak araçların gelişmesi hem de büyük dil modellerinin (LLM) erişilebilir hale gelmesiyle birlikte bu paradigma değişmiştir. Artık güçlü bir dizüstü bilgisayar üzerinde çalışan tek bir araştırmacı, kendi "bilgi grafı laboratuvarını" kurabilir. Bu fikir, veri bilimi, yapay zekâ ve akademik araştırmanın kişiselleşmiş bir birleşimi olarak değerlendirilebilir: tek kişilik bir dijital araştırma ekosistemi.

Bu bölümde, Python, Kùzu ve LLM bileşenleriyle oluşturulabilecek "tek kişilik bilgi grafı laboratuvarı" mimarisi ele alınacaktır.

Laptop Ölçeğinde Yapay Zekâ Destekli Bilgi Sistemleri

Geleneksel olarak bilgi grafı sistemleri büyük sunucular, dağıtık veri tabanları ve kurumsal altyapılar gerektiriyordu. Ancak Kùzu gibi gömülü graf veritabanları ve hafif gömme modelleri sayesinde artık bu sistemler laptop ölçeğine inmiştir.

Bu yeni yaklaşımda:

- Tüm veri yerel diskte saklanır
- Model çalışmaları GPU varsa hızlanır
- İnternet bağlantısı yalnızca veri toplama aşamasında gerekir
- Sistem tamamen kişisel bir araştırma ortamıdır

Bu durum, bireysel araştırmacının veri üzerinde tam kontrol sahibi olmasını sağlar.

Tek Kişilik Araştırma Laboratuvarı Fikri

Tek kişilik bilgi grafı laboratuvarı, aslında bir araştırma ekosistemidir. Bu ekosistem şu bileşenleri içerir:

- Veri toplama araçları
- PDF ve web parsing sistemleri
- Vektör veritabanı
- Graf veritabanı
- Büyük dil modeli (LLM)
- Sorgu ve analiz katmanı

Bu yapı sayesinde bir araştırmacı, kendi akademik evrenini oluşturabilir. Örneğin:

- Tüm PDF makaleleri
- Notlar
- Kitaplar
- Kod parçaları
- Araştırma fikirleri

tek bir sistem içinde birleşir.

Bu yaklaşım, “kişisel bilimsel bilgi grafı” fikrini doğurur.

Python + Kùzu + LLM Mimarisi

Bu laboratuvarın çekirdeği üçlü bir mimariye dayanır:

1. **Python**

- Veri işleme
- Pipeline yönetimi
- API entegrasyonu

2. **Kùzu (graf database)**
    - Varlık ve ilişki depolama
    - Yerel graf sorgular
    - Hızlı traversal işlemleri
3. **LLM (ChatGPT benzeri model)**
    - Doğal dil sorguları
    - Özetleme
    - Soru-cevap üretimi

Bu üçlü yapı birlikte çalışarak şu döngüyü oluşturur:

Veri → Python ile işlenir → Kùzu'ya aktarılır → LLM sorgular → kullanıcıya açıklama döner

Bu mimari, "yerel yapay zekâ laboratuvarı" fikrinin temelidir.

PDF Parsing Pipeline'ı

Bilgi grafı laboratuvarının en önemli veri kaynaklarından biri PDF dosyalarıdır. Akademik makaleler, raporlar ve kitaplar genellikle PDF formatında bulunur.

Pipeline şu adımlardan oluşur:

1. PDF yükleme
2. Metin çıkarımı (PyMuPDF, pdfplumber)
3. Bölüm ayrıştırma (başlık, özet, sonuç)
4. Varlık çıkarımı (entity extraction)
5. İlişki çıkarımı (relation extraction)
6. Graf yapıya dönüştürme

Örneğin bir makalede:

"BERT modeli doğal dil işleme alanında kullanılmaktadır."

Bu cümle şu graf yapısına dönüşür:

- (BERT) —[KULLANILIR]→ (Doğal Dil İşleme)

Bu süreç otomatik hale getirildiğinde, araştırmacı yüzlerce PDF'yi bir bilgi grafına dönüştürebilir.

Gömme Modelleri

Gömme modelleri, metinleri sayısal vektörlere dönüştürerek semantik benzerliği ölçmeyi sağlar. Tek kişilik laboratuvarda gömme katmanı kritik bir rol oynar.

Örneğin:

- "graf neural networks"
- "GNN"
- "graf-based learning"

Bu ifadeler farklı görünse de gömme uzayında birbirine yakın konumlanır.

Bu sayede sistem:

- Benzer makaleleri bulabilir
- Kavramlar arasındaki ilişkiyi anlayabilir
- Semantik arama yapabilir

Local Vector Database Kullanımı

Vektör veritabanı (vector database), gömmelerin saklandığı ve hızlı şekilde sorgulandığı yapıdır. Tek kişilik sistemlerde genellikle şu araçlar kullanılır:

- FAISS
- ChromaDB
- SQLite tabanlı çözümler

Bu sistem sayesinde kullanıcı:

"Benzer makaleleri bul" dediğinde sistem anlam bazlı arama yapar.

Bu yapı özellikle RAG sistemlerinin temel bileşenidir.

Akademik Arşivlerden Veri Toplama

Tek kişilik bilgi grafı laboratuvarı sürekli veri beslenmesine ihtiyaç duyar. Bu nedenle akademik arşivlerden veri toplama kritik bir bileşendir.

Kaynaklar:

- arXiv
- Semantic Scholar

- PubMed
- Google Scholar (dolaylı yöntemler)

Bu veriler otomatik olarak çekilir ve sisteme eklenir.

Kişisel Bilgi Grafı Oluşturma

Tüm bu süreçlerin sonunda kişisel bir bilgi grafı oluşur. Bu graf şunları içerir:

- Okunan makaleler
- Notlar
- Yazarlar
- Konular
- Kavramlar

Örneğin:

- (Kullanıcı) —[OKUDU]→ (Makale A)
- (Makale A) —[KONU]→ (Graf Neural Networks)

Bu yapı zamanla büyüyen bir "kişisel bilimsel hafıza" oluşturur.

Yerel (Local) RAG Sistemleri

Local RAG sistemi, internet bağımsız çalışan bir bilgi üretim mekanizmasıdır. Süreç şu şekilde işler:

1. Kullanıcı soru sorar
2. Sistem yerel graf ve vektör veritabanını tarar
3. İlgili bilgiler seçilir
4. LLM bu bilgileri kullanarak cevap üretir

Bu yapı sayesinde:

- Veri dışarı çıkmaz
- Gizlilik korunur
- Hız artar

Örneğin:

Soru: "GNN nerelerde kullanılır?"

Sistem önce grafı sorgular, sonra LLM açıklama üretir.

Bilimsel Keşif Destek Sistemleri

Bu laboratuvarın en ileri seviyesi, yalnızca bilgi depolamak değil, aynı zamanda yeni fikirler üretmektir.

Sistem şu soruları sorabilir:

- "Bu iki alan arasında bağlantı var mı?"
- "Hangi kavramlar birlikte çalışılmış?"
- "Hangi boşluklar var?"

Bu sayede sistem araştırmacıya şu tür öneriler sunabilir:

- Yeni araştırma konuları
- Disiplinler arası bağlantılar
- Eksik literatür alanları

"Kendi Dijital Araştırma Asistanını" Geliştirmek

Bu sistemin nihai amacı, bireyin kendi dijital araştırma asistanını yaratmasıdır. Bu asistan:

- Tüm akademik geçmişi hatırlar
- Makaleleri analiz eder
- Yeni fikirler önerir
- Soru-cevap üretir
- Literatür taraması yapar

Aslında bu yapı, bir tür "kişisel yapay zekâ araştırma laboratuvarı"dır.

Sonuç

Tek kişilik bilgi grafı laboratuvarı, modern yapay zekâ çağının en güçlü fikirlerinden biridir. Büyük veri merkezlerine ihtiyaç duymadan, tek bir araştırmacı kendi bilgi evrenini kurabilir.

Python, Kùzu ve LLM birleşimi sayesinde:

- Veri anlamlı hale gelir
- Bilgi ilişkisel bir yapıya dönüşür
- Araştırma süreci hızlanır
- Yeni keşifler mümkün olur

Bu yaklaşım, gelecekte araştırmanın bireyselleştiği, her bilim insanının kendi yapay zekâ destekli bilgi grafı sistemine sahip olduğu bir dönemin habercisidir.

9.13 Proje Atölyeleri

Bu bölüm, bilgi grafı yaklaşımının yalnızca teorik bir çerçeve olmadığını, farklı alanlarda doğrudan uygulanabilir güçlü bir analiz aracı olduğunu göstermek için tasarlanmıştır. Burada ele alınan proje atölyeleri, gerçek dünya problemlerinin graf düşünce ile nasıl yeniden modellenebileceğini ortaya koyar. Her bir örnek, düğümler (node), ilişkiler (edge) ve semantik katmanlar üzerinden karmaşık sistemleri daha anlaşılır ve analiz edilebilir hale getirir.

Sahte Haber Analiz Ağı

Sahte haber (fake news) günümüz bilgi ekosisteminin en kritik problemlerinden biridir. Bu problemi graf tabanlı modellemek, bilgi yayılımını anlamak açısından son derece etkilidir.

Bir sahte haber analiz ağında şu bileşenler bulunur:

- Haber kaynakları (düğüm)
- Makaleler (düğüm)
- Sosyal medya paylaşımları (düğüm)
- Kullanıcılar (düğüm)
- Paylaşım ilişkileri (bağlantı)
- Alıntı ve referans bağlantıları (bağlantı)

Örneğin bir sahte haber, belirli bir kaynaktan çıkar ve sosyal medya üzerinden hızla yayılır. Bu yayılım graf üzerinde bir “yayılma zinciri” (propagation chain) olarak görülebilir.

Graf analizi sayesinde:

- Hangi kaynakların daha fazla sahte bilgi ürettiği
- Bilginin hangi düğümler üzerinden yayıldığı
- Hangi kullanıcıların yanlış bilgi yayılımında merkez olduğu

tespit edilebilir. Özellikle merkeziyet (centrality) analizleri, sahte haberin yayılımındaki kritik düğümleri ortaya çıkarır.

Finansal Dolandırıcılık İlişkileri

Finansal dolandırıcılık, karmaşık ilişkiler ağı üzerinden gerçekleştirilen bir suç türüdür. Bilgi grafı bu karmaşık yapıyı çözmek için güçlü bir araçtır.

Bu ağda şu varlıklar bulunur:

- Hesaplar
- Şirketler
- Bankalar

- Transfer işlemleri
- IP adresleri
- Kullanıcı kimlikleri

İlişkiler ise şunları temsil eder:

- Para transferi
- Ortak hesap kullanımı
- Aynı IP üzerinden giriş
- Şüpheli işlem bağlantıları

Örneğin:

- (Kullanıcı A) —[TRANSFER YAPTI]→ (Kullanıcı B)
- (Kullanıcı B) —[AYNI IP]→ (Kullanıcı C)

Bu yapı sayesinde gizli dolandırıcılık ağları ortaya çıkarılabilir. Özellikle döngüsel yapılar (cycles) ve anormal bağlantı yoğunlukları, dolandırıcılık tespiti için kritik sinyaller üretir.

Sosyal Medya Etkileşim Ağı

Sosyal medya platformları doğal olarak devasa bir graf yapıdır. Kullanıcılar, paylaşımlar ve etkileşimler bu grafın temel bileşenleridir.

Düğümler:

- Kullanıcılar
- Gönderiler
- Hashtag’ler
- Yorumlar

İlişkiler:

- Takip etme
- Beğenme
- Paylaşma
- Yorum yapma

Örneğin:

- (Kullanıcı A) —[FOLLOWS]→ (Kullanıcı B)
- (Kullanıcı A) —[LIKE]→ (Gönderi X)

Bu ağ üzerinden:

- Viral içerikler tespit edilebilir
- Etkileşim merkezleri belirlenebilir
- Topluluk yapıları analiz edilebilir

Ayrıca sosyal medya grafı, bilgi yayılımının dinamiklerini incelemek için de kullanılır.

Patent ve İnovasyon Ağları

Patent sistemleri, teknolojik gelişimin izlenmesi için zengin bir graf veri kaynağıdır. Her patent bir düğüm, patentler arasındaki atıflar ise ilişkidir.

Düğümler:

- Patentler
- Buluşçular
- Şirketler
- Teknoloji alanları

İlişkiler:

- Atıf yapma
- Ortak geliştirme
- Aynı teknoloji alanına ait olma

Örneğin:

- (Patent A) —[ATIF]→ (Patent B)
- (Şirket X) —[SAHİP]→ (Patent A)

Bu yapı sayesinde:

- Teknolojik trendler analiz edilebilir
- İnovasyon merkezleri belirlenebilir
- Gelecekteki teknoloji yönelimleri tahmin edilebilir

Patent grafı aynı zamanda Ar-Ge stratejilerinin belirlenmesinde önemli bir araçtır.

Tedarik Zinciri Grafı

Tedarik zinciri, üretimden tüketiciye kadar uzanan karmaşık bir ilişkiler ağından oluşur. Bu yapı doğal olarak graf modelleme için uygundur.

Düğümler:

- Tedarikçiler
- Fabrikalar
- Depolar
- Ürünler
- Müşteriler

İlişkiler:

- Malzeme tedarik etme
- Üretim sürecine katılma
- Dağıtım yapma

Örneğin:

- (Tedarikçi A) —[SAĞLAR]→ (Fabrika B)
- (Fabrika B) —[ÜRETİR]→ (Ürün C)

Bu graf yapı sayesinde:

- Tedarik zinciri riskleri analiz edilebilir
- Darboğazlar (bottlenecks) tespit edilebilir
- Lojistik optimizasyon yapılabilir

Özellikle küresel krizlerde tedarik zinciri grafı kritik bir analiz aracıdır.

Bilim İnsanları İlişki Haritası

Bilim dünyası, aslında dev bir sosyal ağdır. Bilim insanları arasındaki işbirlikleri, atıflar ve ortak projeler bu ağı oluşturur.

Düğümler:

- Bilim insanları
- Üniversiteler
- Araştırma projeleri
- Yayınlar

İlişkiler:

- Ortak makale yazma
- Atıf yapma

- Aynı projede çalışma

Örneğin:

- (Bilim İnsanı A) —[ORTAK YAYIN]→ (Bilim İnsanı B)
- (Bilim İnsanı A) —[ÇALIŞIR]→ (Üniversite X)

Bu ağ üzerinden:

- Bilimsel işbirliği merkezleri belirlenebilir
- Disiplinler arası bağlantılar analiz edilebilir
- Araştırma toplulukları haritalanabilir

Bu tür analizler, bilim politikalarının oluşturulmasında da önemli rol oynar

Sonuç

Bu proje atölyeleri, bilgi grafı yaklaşımının çok farklı alanlara nasıl uygulanabileceğini göstermektedir. Sahte haber analizi, finansal dolandırıcılık tespiti, sosyal medya analizi, patent haritaları, tedarik zinciri modellemesi ve bilim insanı ağları; hepsi aslında aynı temel yapıya dayanır: varlıklar ve ilişkiler ağı.

Graf düşünce, bu karmaşık sistemleri sadeleştirerek görünür hale getirir. Böylece yalnızca veri değil, veri arasındaki anlamlı bağlantılar da analiz edilebilir. Bu yaklaşım, modern veri bilimi ve yapay zekânın en güçlü analiz araçlarından biri haline gelmiştir.

9.14 Bölüm Sonu: Koddan Anlama

Bu bölüm, yalnızca teknik bir yolculuğun değil, aynı zamanda düşünme biçiminin dönüşümünün de bir özeti olarak ele alınmalıdır. Veri tabanlarından bilgi grafı sistemlerine, SQL sorgularından semantik modellere ve oradan yapay zekâ destekli anlam üretimine uzanan bu süreç, aslında "kod yazma" eyleminden "anlam kurma" düzeyine geçişi temsil eder. Artık mesele yalnızca veriyi saklamak veya sorgulamak değil; verinin içindeki ilişkileri, bağlamı ve anlamı ortaya çıkarmaktır.

Graf Düşüncesinin Dönüşümü

Graf düşüncesi, modern veri biliminin ve yapay zekânın en temel zihinsel modellerinden biri haline gelmiştir. Geleneksel yaklaşımlar dünyayı satırlar, sütunlar ve tablolar üzerinden anlamaya çalışırken, graf yaklaşımı dünyayı düğümler ve ilişkiler üzerinden okur.

Bu dönüşüm basit bir teknik değişim değildir; bilişsel bir paradigmadır. Çünkü insan zihni de aslında ilişkisel düşünür. Bir kişiyi tanımlarken yalnızca kim olduğunu değil, kimlerle bağlantılı olduğunu, hangi olaylara dahil olduğunu ve hangi bağlamlarda bulunduğunu düşünür.

Graf düşüncesinin yükselişi şunu göstermiştir:

- Veri izole değildir
- Bilgi bağlantılarla anlam kazanır
- Gerçek dünya lineer değil ağsaldır

Bu nedenle graf düşüncesi, yalnızca bir veri modeli değil, bir “dünya okuma biçimi”dir.

SQL’den Semantik Düşünceye Geçiş

SQL (Structured Query Language), uzun yıllar boyunca veri tabanlarının temel sorgu dili olmuştur. Ancak SQL’in temel varsayımı, verinin tablolarda düzenli ve sabit şemalarla saklanmasıdır. Bu yaklaşım, ilişkilerin ikinci planda kaldığı bir veri modeline dayanır.

Oysa modern veri problemleri çok daha karmaşıktır:

- Dinamik ilişkiler
- Çok katmanlı bağlantılar
- Değişken veri yapıları

Bu noktada semantik düşünce devreye girer. Semantik yaklaşımda veri yalnızca “ne olduğu” ile değil, “ne anlama geldiği” ile ilgilidir.

Örneğin SQL’de bir sorgu:

SELECT * FROM users WHERE age > 30;

Bu sorgu yalnızca bir filtreleme işlemi yapar. Ancak semantik bir sistemde soru şu hale gelir:

“30 yaşından büyük ve belirli bir davranış örüntüsüne sahip kullanıcılar kimlerdir?”

Bu dönüşüm, veriden anlam üretmeye geçişin temelini oluşturur. Graf veri tabanları ve bilgi grafı sistemleri bu semantik dönüşümün teknik karşılığıdır.

Bilgi Grafı Mühendisliğinin Geleceği

Bilgi grafı mühendisliği, gelecekte veri mühendisliğinin en kritik alanlarından biri haline gelecektir. Çünkü modern yapay zekâ sistemleri yalnızca büyük veri ile değil, yapılandırılmış anlam ağları ile daha iyi çalışmaktadır.

Gelecekte bilgi grafı mühendisliğinin öne çıkacağı alanlar:

- Yapay zekâ destekli karar sistemleri
- Kurumsal veri entegrasyonu
- Sağlık bilgi sistemleri
- Bilimsel keşif platformları
- Otonom sistemler

Bu alandaki temel dönüşüm şudur: veri mühendisliği artık sadece “veriyi taşımak” değil, “veriyi anlamlandırmak” haline gelmektedir.

Bilgi grafı mühendisleri gelecekte şu becerilere sahip olacaktır:

- Ontoloji tasarımı
- Semantik modelleme
- LLM entegrasyonu
- Graf algoritmaları
- Veri yönetişimi

Bu disiplin, veri bilimi ile yapay zekâ mühendisliği arasında köprü kuran yeni bir alan olarak gelişmektedir.

Açık Kaynak Graf Ekosisteminin Yükselişi

Bilgi grafı teknolojilerinin gelişiminde açık kaynak ekosisteminin rolü çok büyüktür. Neo4j, Kùzu, NetworkX, Apache Jena, RDF4J gibi sistemler bu ekosistemin önemli parçalarıdır.

Açık kaynak yaklaşımı sayesinde:

- Graf veri tabanları demokratikleşmiştir
- Küçük ekipler güçlü sistemler kurabilmektedir
- Araştırma ve geliştirme hızlanmıştır
- Deneysel projeler kolaylaşmıştır

Özellikle Kùzu gibi gömülü graf sistemleri, bireysel araştırmacıların kendi bilgisayarlarında büyük ölçekli graf analizler yapabilmesini sağlamıştır.

Açık kaynak ekosisteminin yükselişi şu sonucu doğurmuştur:

Bilgi grafı artık yalnızca büyük şirketlerin değil, bireysel araştırmacıların da aracı haline gelmiştir.

Yapay Zekâ Çağında Bireysel Araştırmacının Güçlenmesi

Yapay zekâ çağının en önemli dönüşümlerinden biri, bireysel araştırmacının güçlenmesidir. Geçmişte büyük veri setlerine erişim, yüksek hesaplama gücü ve kurumsal altyapılar olmadan ileri düzey araştırma yapmak zordu. Ancak bugün bu durum değişmiştir.

Artık bir araştırmacı:

- LLM modelleri kullanabilir
- Yerel graf veritabanı kurabilir

- PDF ve web verilerini otomatik işleyebilir
- Vektör veritabanı ile semantik arama yapabilir

Bu araçların birleşimi, tek kişilik güçlü araştırma laboratuvarları oluşturmayı mümkün kılar.

Bu dönüşüm şunu ifade eder:

Bilgi üretimi artık merkezi değildir; dağıtık ve bireysel hale gelmiştir.

Bir araştırmacı kendi bilgi grafını oluşturduğunda:

- Kendi literatürünü yönetebilir
- Kendi keşif sistemini kurabilir
- Kendi yapay zekâ asistanını geliştirebilir

Bu durum, bilimsel üretimi hızlandıran yeni bir çağın başlangıcıdır.

Koddan Anlama Geçişi

Bu bölümün en önemli mesajı şudur: artık kod yazmak tek başına yeterli değildir; önemli olan anlam üretmektir.

Eskiden yazılım geliştirme şu şekildeydi:

- Veri girilir
- İşlenir
- Çıktı üretilir

Bugün ise süreç farklıdır:

- Veri anlamlandırılır
- İlişkiler modellenir
- Yapay zekâ ile yorumlanır
- Bilgi sürekli evrilir

Bu nedenle modern geliştirici artık sadece “kodlayıcı” değil, aynı zamanda “bilgi mühendisi”dir. Kod, artık yalnızca bir araçtır. Asıl hedef, verinin içindeki anlamı açığa çıkarmaktır.

Sonuç

“Koddan Anlama” geçişi, bilgi grafı çağının özünü oluşturur. SQL tabanlı veri dünyasından semantik graf dünyasına geçiş, yalnızca teknik bir evrim değil, düşünsel bir devrimdir.

Bu dönüşümle birlikte:

- Veri ilişkisel hale gelir

- Bilgi ağsallaşır
- Yapay zekâ daha açıklanabilir olur
- Araştırma süreçleri hızlanır
- Bireysel üretkenlik artar

Sonuç olarak bilgi grafı mühendisliği, yalnızca bir teknoloji alanı değil, yeni bir düşünme biçimidir. Bu düşünme biçimi, veriyi anlamlandırarak onu bilgiye, bilgiyi de keşfe dönüştürür.

BÖLÜM 10 — GELECEK

Bilgi grafı teknolojileri, yapay zekâ ile birleştiğinde yalnızca bir veri modelleme yaklaşımı olmaktan çıkar; insanlığın bilgiyle kurduğu ilişkinin yeni bir biçimine dönüşür. Bu nedenle "gelecek" başlığı, teknik bir öngörüden çok daha fazlasını ifade eder: bilginin nasıl üretileceği, nasıl saklanacağı, nasıl paylaşılacağı ve en önemlisi nasıl anlamlandırılacağı sorularının yeniden yazıldığı bir dönemi anlatır.

Gelecekte veri sistemleri artık tablo merkezli veya belge merkezli yapılardan ziyade, tamamen ilişki merkezli yapılara evrilecektir. Çünkü gerçek dünya doğası gereği ağsaldır. İnsanlar, kurumlar, makineler, bilimsel kavramlar ve olaylar birbirinden bağımsız varlıklar değil; sürekli etkileşim halinde olan düğümlerden oluşan dev bir sistemdir. Bilgi grafı bu yapıyı doğal biçimde temsil eder ve bu nedenle geleceğin bilgi mimarilerinin merkezine yerleşir.

Bu dönüşümün en kritik bileşeni yapay zekâdır. Büyük dil modelleri (LLM'ler), yalnızca metin üretmekle sınırlı kalmayıp, bilgi grafı sistemleriyle entegre edildiğinde "anlam tabanlı akıl yürütme" yeteneği kazanır. Bu noktada geleceğin yapay zekâ sistemleri yalnızca cevap veren sistemler değil, aynı zamanda bağlam kuran, ilişkileri keşfeden ve eksik bilgiyi tamamlayan sistemler haline gelir. GrafRAG gibi yaklaşımlar, bu dönüşümün erken örnekleridir.

Gelecekte bilgi grafı sistemleri yalnızca akademik veya kurumsal alanlarda değil, günlük yaşamın her alanında görünmez bir altyapı haline gelecektir. Akıllı şehirler, sağlık sistemleri, finansal altyapılar ve hatta bireysel dijital asistanlar, arka planda sürekli güncellenen bilgi grafı yapıları üzerinde çalışacaktır. Bu durum, insan-makine etkileşimini daha doğal, daha bağlamsal ve daha sezgisel hale getirecektir.

Bir diğer önemli dönüşüm ise bireyselleşmedir. Gelecekte her birey, kendi kişisel bilgi grafına sahip olacaktır. Bu graf yapı; okunan kitapları, izlenen makaleleri, yazılan notları, yapılan konuşmaları ve öğrenilen kavramları birbirine bağlayan yaşayan bir hafıza sistemi olacaktır. Böylece bilgi artık yalnızca dış kaynaklarda değil, bireyin kendi dijital zihninde de örgütlenmiş bir yapı haline gelecektir.

Açık kaynak ekosisteminin büyümesi, bu dönüşümü daha da hızlandıracaktır. Graf veri tabanları, gömme modelleri ve yerel yapay zekâ sistemleri demokratikleşerek herkesin erişimine açık hale gelecektir. Bu da bilgi üretimini merkezsizleştirecek ve daha geniş bir katılıma açacaktır.

Sonuç olarak "Gelecek", yalnızca daha hızlı bilgisayarlar veya daha büyük modeller anlamına gelmez. Gelecek, bilginin doğasının değiştiği bir dönemdir. Veriden anlam üretme süreci, insan zihni ile yapay zekâ arasında ortak bir düşünme alanı oluşturacaktır. Bilgi grafı bu alanın

omurgasını oluştururken, yapay zekâ bu omurganın üzerinde çalışan dinamik bir akıl haline gelecektir.

10.1 Web 3.0 ve Semantik İnternet

İnternetin gelişim süreci genellikle üç büyük evre üzerinden açıklanır: Web 1.0, Web 2.0 ve Web 3.0. Web 1.0 dönemi, kullanıcıların yalnızca içerik okuduğu statik sayfalardan oluşuyordu. Web 2.0 ise sosyal medya, kullanıcı üretimli içerik ve etkileşimli platformların yükselişini temsil etti. Ancak bugün giderek daha fazla konuşulan Web 3.0 kavramı, internetin yalnızca insanlar tarafından okunabilen değil, makineler tarafından da "anlaşılabilen" bir yapıya dönüşmesini ifade eder. Bu dönüşümün merkezinde ise semantik internet (semantic web) yaklaşımı yer alır.

Semantik internet fikrinin temel amacı, internetteki verilerin yalnızca metin olarak değil, anlam ilişkileriyle birlikte temsil edilmesidir. Günümüzde web üzerindeki büyük veri miktarı çoğunlukla yapılandırılmamış durumdadır. Bir web sayfası insanlar için anlamlı olabilir; ancak makineler açısından çoğu zaman yalnızca karakter dizilerinden oluşur. Semantik internet yaklaşımı ise web verilerini bilgi grafı mantığıyla yapılandırarak makinelerin içerikleri anlamlandırabilmesini hedefler.

Örneğin klasik bir web sayfasında "Albert Einstein fizikçidir" ifadesi yalnızca düz metin olarak bulunabilir. Semantik internet yaklaşımında ise bu bilgi şu şekilde temsil edilir:

- Albert Einstein → MESLEK → Fizikçi

Bu yapı sayesinde yapay zekâ sistemleri yalnızca kelimeleri değil, ilişkileri de anlayabilir. Böylece internet, birbirinden kopuk sayfaların koleksiyonu olmaktan çıkıp dev bir bilgi ağına dönüşür.

Web 3.0 yaklaşımında RDF (Resource Description Framework), OWL (Web Ontology Language) ve SPARQL gibi teknolojiler önemli rol oynar. RDF, bilgiyi özne–yüklem–nesne üçlüleriyle temsil ederken; OWL kavramlar arasındaki mantıksal ilişkileri tanımlar. SPARQL ise bu semantik veri üzerinde sorgulama yapılmasını sağlar. Bu teknolojiler birlikte çalışarak "makine tarafından okunabilir internet" fikrini mümkün hale getirir.

Semantik internetin en önemli sonuçlarından biri bağlamsal aramadır. Günümüzde arama motorları büyük ölçüde anahtar kelime eşleşmelerine dayanır. Ancak gelecekte arama sistemleri kullanıcı niyetini, kavramsal ilişkileri ve bağlamı çok daha derin biçimde analiz edecektir. Örneğin kullanıcı "Türkiye'de Nobel ödülü alan bilim insanları" diye arama yaptığında sistem yalnızca kelime eşleşmesi yapmayacak; "Nobel", "bilim insanı", "Türkiye" ve "ödül" kavramları arasındaki ilişkileri bilgi grafı üzerinden yorumlayacaktır.

Web 3.0 aynı zamanda merkeziyetsizlik (decentralization) fikriyle de ilişkilidir. Büyük teknoloji şirketlerinin veri tekeline karşı daha dağıtık yapılar geliştirilmesi hedeflenmektedir. Blockchain teknolojileri, dağıtık kimlik sistemleri ve semantik veri yapıları birlikte çalışarak kullanıcıların verileri üzerinde daha fazla kontrol sahibi olduğu yeni bir internet mimarisi oluşturabilir.

Yapay zekâ açısından bakıldığında semantik internet, LLM sistemlerinin daha güvenilir ve açıklanabilir hale gelmesini sağlayabilir. Büyük dil modelleri bugün çoğunlukla olasılık tabanlı metin üretimi yapmaktadır. Ancak semantik bilgi katmanlarıyla birleştirildiklerinde, daha tutarlı, doğrulanabilir ve bağlamsal cevaplar üretebilirler.

Sonuç olarak Web 3.0 ve semantik internet, internetin yeni evrim aşamasını temsil eder. Bu yeni aşamada internet yalnızca bilgi depolayan bir ortam değil, anlam üreten ve ilişkileri anlayan küresel bir bilgi organizmasına dönüşmektedir. Bilgi grafı teknolojileri ise bu organizmanın sinir sistemi gibi çalışmaktadır.

10.2 Dijital Hafıza Sistemleri

İnsanlık tarihi boyunca hafıza, bilginin korunmasının ve aktarılmasının temel aracı olmuştur. Sözlü kültür döneminde hafıza tamamen insan zihnine dayanıyordu. Yazının icadıyla birlikte bilgi dışsallaştırıldı; kitaplar, arşivler ve kütüphaneler insanlığın kolektif belleği haline geldi. Dijital çağda ise yeni bir aşamaya geçiyoruz: dijital hafıza sistemleri.

Dijital hafıza sistemleri, bireylerin veya kurumların bilgi birikimini dijital ortamda yapılandırılmış biçimde saklayan, ilişkilendiren ve gerektiğinde yeniden erişilebilir hale getiren sistemlerdir. Ancak geleceğin dijital hafızası yalnızca belge depolayan bir arşiv olmayacaktır. Bilgi grafı ve yapay zekâ birleşimi sayesinde bu sistemler aktif düşünsel yardımcılar haline dönüşecektir.

Bugün insanlar büyük miktarda bilgiyle çevrilidir:

- PDF makaleler
- Web sayfaları
- E-postalar
- Notlar
- Videolar
- Ses kayıtları

Ancak bu bilgi çoğu zaman dağınık durumdadır. Dijital hafıza sistemlerinin amacı, bu parçalanmış veriyi ilişkisel bir ağ içinde birleştirmektir.

Örneğin bir araştırmacının dijital hafızası şu ilişkileri içerebilir:

- Okuduğu makaleler
- Yazdığı notlar
- İlgilendiği kavramlar
- Çalıştığı kişiler

- Geliştirdiği projeler

Bu yapı zamanla bireyin düşünsel haritasına dönüşür.

Bilgi grafı teknolojileri burada kritik rol oynar. Çünkü dijital hafıza yalnızca veri depolamak değil, veriler arasındaki anlamlı bağlantıları kurmak anlamına gelir. Örneğin bir sistem şunu anlayabilir:

- "Bu makale daha önce okuduğun başka bir çalışma ile ilişkili."
- "Bu kavram geçen yıl aldığın notlarda da geçiyordu."
- "Bu araştırma konusu ilgilendiğin başka bir alanla bağlantılı."

Bu yaklaşım, hafızayı pasif bir arşiv olmaktan çıkarıp aktif bir düşünce sistemine dönüştürür.

Yapay zekâ destekli dijital hafıza sistemleri gelecekte kişisel araştırma asistanları gibi çalışacaktır. Kullanıcı doğal dilde soru sorabilecek:

- "Geçen yıl kuantum hesaplama hakkında hangi makaleleri okumuştum?"
- "Bu kavramla ilişkili notlarımı göster."
- "Bu araştırma fikrine benzer çalışmalar var mı?"

Sistem bilgi grafı ve gömme yapıları üzerinden cevap üretecektir.

Kurumsal düzeyde dijital hafıza sistemleri ise organizasyonel bilgi kaybını azaltabilir. Şirketlerde çalışanlar ayrıldığında büyük miktarda örtük bilgi kaybolur. Bilgi grafı tabanlı kurumsal hafıza sistemleri, bu bilgiyi yapısal biçimde koruyabilir.

Gelecekte dijital hafıza sistemleri artırılmış gerçeklik (AR), giyilebilir cihazlar ve sürekli çalışan yapay zekâ ajanlarıyla birleşebilir. Böylece bireyin dijital hafızası, günlük yaşamın görünmez bir katmanı haline gelebilir.

Ancak bu gelişmeler önemli etik soruları da beraberinde getirir:

- Kim hangi hafızaya erişebilir?
- Dijital unutma mümkün olacak mı?
- Hafızanın manipülasyonu nasıl önlenecek?
- İnsan hafızası ile yapay hafıza arasındaki sınır ne olacak?

Bu sorular geleceğin en önemli tartışma alanlarından biri olacaktır.

Sonuç olarak dijital hafıza sistemleri, insanlığın bilgiyle ilişkisinde yeni bir evreyi temsil eder. Bilgi grafı, semantik teknolojiler ve yapay zekâ birleşerek insan zihninin dijital uzantıları haline gelen yeni nesil hafıza yapıları oluşturmaktadır.

10.3 İnsanlığın Ortak Bilgi Grafı

İnsanlık tarihi, aynı zamanda bilginin örgütlenme tarihidir. Mağara resimlerinden papirüslere, kütüphanelerden dijital veri merkezlerine kadar her dönem, insanlığın bilgi üretme ve bilgiyi koruma biçimini yeniden tanımlamıştır. Ancak bugün ortaya çıkan bilgi miktarı, tarihte görülmemiş bir büyüklüğe ulaşmıştır. Bilimsel makaleler, sosyal medya içerikleri, sensör verileri, videolar, kitaplar ve dijital belgeler sürekli olarak çoğalmaktadır. Bu nedenle geleceğin en önemli problemlerinden biri, insanlığın ürettiği bu devasa bilgi okyanusunun nasıl anlamlı bir yapıya dönüştürüleceğidir. "İnsanlığın ortak bilgi grafı" fikri tam olarak bu noktada ortaya çıkar.

Ortak bilgi grafı, insanlığın ürettiği bilgilerin birbirine bağlandığı küresel bir anlam ağıdır. Bu yapı yalnızca belgeleri depolamaz; insanlar, kurumlar, bilimsel kavramlar, tarihsel olaylar, teknolojiler ve kültürel üretimler arasındaki ilişkileri de temsil eder. Böylece dünya, birbirinden kopuk veri kümelerinden oluşan bir arşiv olmaktan çıkar ve devasa bir ilişkiler ağına dönüşür.

Bu yaklaşımın erken örnekleri bugün zaten görülmektedir. Arama motorlarının kullandığı bilgi grafı sistemleri, Wikipedia bağlantıları, Wikidata, bilimsel atıf ağları ve sosyal ağ platformları aslında bu büyük yapının parçalarıdır. Ancak gelecekte bu sistemlerin çok daha bütünleşik hale gelmesi beklenmektedir.

Örneğin gelecekte bir araştırmacı şu soruyu sorabilir:

"19. yüzyıl fizik düşüncesi ile modern yapay zekâ arasındaki dolaylı bağlantılar nelerdir?"

Bu soruya cevap verebilmek için sistemin yalnızca belgeleri değil, fikirler arasındaki tarihsel ilişkileri de anlaması gerekir. İşte ortak bilgi grafı bu tür bağlantıları mümkün kılar.

Bu yapı aynı zamanda insanlığın kolektif hafızası olarak düşünülebilir. Çünkü bilgi grafı yaklaşımı, yalnızca bilgiyi depolamak değil, bilgiyi bağlamı içinde korumak anlamına gelir. Bir bilimsel keşif yalnızca tek bir makale değildir; o keşif başka araştırmalara, kişilere, kurumlara ve tarihsel olaylara bağlıdır. Ortak bilgi grafı bu ilişkileri görünür hale getirir.

Yapay zekâ açısından bakıldığında ortak bilgi grafı, geleceğin küresel bilişsel altyapısı olabilir. Büyük dil modelleri bugün devasa metin koleksiyonlarından öğrenmektedir; ancak bu öğrenme çoğu zaman istatistiksel örüntülere dayanır. Bilgi grafı sistemleriyle birleştiğinde ise yapay zekâ, yalnızca kelime olasılıklarını değil, anlam ilişkilerini de kullanabilir.

Bu durum özellikle bilimsel keşif süreçlerinde önemlidir. Gelecekte yapay zekâ sistemleri:

- Farklı disiplinler arasındaki bağlantıları keşfedebilir
- Daha önce ilişkilendirilmemiş kavramları bir araya getirebilir
- Bilimsel boşlukları tespit edebilir
- Yeni araştırma hipotezleri önerebilir

Böylece insanlığın ortak bilgi grafı, yalnızca geçmiş bilginin deposu değil, yeni bilginin üretim motoru haline gelir.

Ancak bu fikir önemli sorunları da beraberinde getirir. Ortak bilgi grafını kim yönetecektir? Hangi bilgiler doğru kabul edilecektir? Bilgi üzerindeki politik ve ekonomik güç dengeleri nasıl etkilenecektir? Eğer küresel bilgi birkaç büyük şirketin kontrolüne girerse, bu yapı dijital bir epistemolojik tekele dönüşebilir.

Bu nedenle açık kaynak ve dağıtık bilgi sistemleri kritik önem taşır. Wikidata gibi topluluk tabanlı yapılar, ortak bilgi grafının demokratik biçimde gelişebilmesi için önemli örneklerdir.

Sonuç olarak insanlığın ortak bilgi grafı, internetin gelecekteki evriminin merkezinde yer alabilir. Bu yapı, insan bilgisini yalnızca depolayan değil; ilişkilendiren, açıklayan ve sürekli yeniden organize eden küresel bir bilişsel ağ olacaktır. Belki de geleceğin uygarlıkları, bugün şehirleri ve yolları nasıl düşünüyorsak, bilgi graflarını da insanlığın görünmez altyapısı olarak öyle düşünecektir.

10.4 Yapay Genel Zekâ ve Bilgi Temsili

Yapay zekâ alanındaki en büyük hedeflerden biri Yapay Genel Zekâdır (Artificial General Intelligence – AGI). Yapay Genel Zekâ, yalnızca belirli görevlerde başarılı olan sistemleri değil, insan benzeri genel düşünme ve öğrenme yeteneğine sahip sistemleri ifade eder. Bugünkü yapay zekâ modelleri etkileyici sonuçlar üretse de çoğu hâlâ dar uzmanlık alanlarında çalışmaktadır. Oysa gerçek bir AGI sistemi, farklı alanlar arasında bilgi transferi yapabilmeli, bağlam anlayabilmeli ve soyut akıl yürütme gerçekleştirebilmelidir. Bu noktada "bilgi temsili" problemi merkezi bir rol oynar.

Bilgi temsili (knowledge representation), bir makinenin dünyayı nasıl modellediği sorusuyla ilgilidir. İnsan zihni yalnızca veri depolamaz; kavramlar arasında ilişkiler kurar, neden-sonuç bağları oluşturur ve bağlama göre yorum yapar. Eğer bir yapay zekâ sistemi insan benzeri düşünme kapasitesine ulaşacaksa, bilgiyi benzer biçimde temsil edebilmelidir.

Klasik yapay zekâ çalışmalarında bilgi temsili çoğunlukla sembolik sistemlere dayanıyordu. Mantık kuralları, ontolojiler ve uzman sistemler aracılığıyla bilgi açık biçimde modellenmeye çalışılıyordu. Örneğin:

- İnsan → canlıdır
- Kuş → uçabilir
- Penguen → kuştur ama uçamaz

Bu sistemler açıklanabilir ve mantıksal olarak tutarlıydı; ancak gerçek dünyanın karmaşıklığını ölçeklendirmekte zorlanıyordu.

Modern büyük dil modelleri ise farklı bir yaklaşım kullanır. Bu sistemler bilgiyi açık kurallar yerine devasa parametre uzaylarında dağıtık biçimde temsil eder. Başka bir deyişle bilgi, sinir

ağı ağırlıklarına gömülmüş durumdadır. Bu yaklaşım olağanüstü esneklik sağlar; ancak önemli bir sorun ortaya çıkarır: bilgi açık biçimde temsil edilmediği için sistemin neden belirli bir cevap verdiğini anlamak zorlaşır.

İşte bilgi grafı sistemleri burada yeniden önem kazanır. Çünkü bilgi grafı yaklaşımı, kavramlar arasındaki ilişkileri açık ve izlenebilir biçimde temsil eder. AGI'ye giden yolda birçok araştırmacı, yalnızca büyük dil modellerinin yeterli olmayacağını; sembolik bilgi yapılarıyla birleşmeleri gerektiğini düşünmektedir.

Bu yaklaşım genellikle "sinirsel-sembolik yapay zekâ" (neuro-symbolic YZ) olarak adlandırılır. Burada:

- Sinir ağları örüntü öğrenir
- Bilgi grafı ilişkileri temsil eder
- Mantık sistemleri akıl yürütme sağlar

Bu birleşim, gelecekte daha güçlü AGI sistemlerinin temelini oluşturabilir.

Örneğin bir AGI sistemi şu tür çıkarımlar yapabilmelidir:

- "Bir ilaç belirli bir proteini etkiliyorsa, bu proteinle ilişkili hastalıkları da etkileyebilir."
- "Bir bilimsel teori başka bir teoriyle çelişiyorsa, hangi bağlamlarda geçerli oldukları incelenmelidir."

Bu tür akıl yürütmeler yalnızca istatistiksel örüntülerle değil, yapılandırılmış bilgi temsiliyle mümkündür.

Bilgi grafı burada insan zihnine benzer bir "kavramsal harita" görevi görür. Çünkü insanlar da bilgiyi çoğu zaman ilişkisel ağlar halinde organize eder. Hafıza, çağrışımlar ve kavramsal bağlantılar aslında biyolojik bir bilgi grafı gibi çalışır.

AGI bağlamında bir diğer kritik konu ise sürekli öğrenmedir. İnsan zihni yeni bilgi geldikçe mevcut bilgi yapısını günceller. Geleceğin yapay zekâ sistemleri de dinamik bilgi grafı yapılarıyla sürekli evrilen bilişsel ağlar geliştirebilir.

Ancak burada felsefi bir soru ortaya çıkar: "Bilgiyi temsil etmek, anlamak anlamına gelir mi?" Bir sistem milyonlarca ilişkiyi depolayabilir; fakat gerçekten "anlıyor" mudur? Bu soru AGI tartışmalarının merkezindeki problemlerden biridir.

Sonuç olarak Yapay Genel Zekâ'nın geleceği büyük ölçüde bilgi temsili problemine bağlıdır. Bilgi grafı sistemleri, ontolojiler ve semantik yapılar; yapay zekânın yalnızca metin üreten değil, dünyayı kavramsal olarak modelleyebilen sistemlere dönüşmesinde kritik rol oynayacaktır. Bu nedenle AGI'ye giden yol, yalnızca daha büyük modellerden değil, daha derin anlam temsillerinden geçmektedir.

10.5 Bilginin Politik Ekonomisi

Bilgi, insanlık tarihi boyunca yalnızca kültürel veya bilimsel bir değer değil, aynı zamanda ekonomik ve politik bir güç kaynağı olmuştur. Tarım toplumlarında toprak, sanayi çağında üretim araçları ne kadar önemliyse, dijital çağda da bilgi aynı ölçüde stratejik hale gelmiştir. Bugün dünyanın en güçlü şirketleri petrol, çelik veya fiziksel hammaddeler üzerinden değil; veri, algoritma ve bilgi ağları üzerinden büyümektedir. Bu nedenle modern dünyayı anlamak için "bilginin politik ekonomisi" kavramı giderek daha önemli hale gelmektedir.

Politik ekonomi yaklaşımı, ekonomik sistemlerin yalnızca piyasa dinamikleriyle değil, aynı zamanda güç ilişkileriyle şekillendiğini savunur. Bilgi ekonomisi de benzer biçimde çalışır. Çünkü hangi bilginin üretileceği, hangi bilginin görünür olacağı ve hangi bilginin erişilebilir olacağı çoğu zaman ekonomik ve politik güç yapıları tarafından belirlenir.

Dijital çağda veri yeni bir sermaye biçimine dönüşmüştür. Sosyal medya platformları, arama motorları, e-ticaret şirketleri ve yapay zekâ firmaları milyarlarca insanın davranışsal verisini toplamaktadır. Bu veriler yalnızca reklam hedefleme amacıyla kullanılmaz; aynı zamanda insan davranışlarını tahmin etmek, yönlendirmek ve optimize etmek için de kullanılır.

Bilgi grafı sistemleri bu süreçte kritik rol oynar. Çünkü ham veri tek başına sınırlı değere sahiptir; asıl güç, veriler arasındaki ilişkilerin modellenmesiyle ortaya çıkar. Örneğin bir kullanıcının:

- Arama geçmişi
- Konum bilgisi
- Sosyal ilişkileri
- Alışveriş davranışları
- İlgi alanları

tek başına farklı veri parçalarıdır. Ancak bilgi grafı yaklaşımıyla bunlar ilişkilendirildiğinde son derece ayrıntılı davranış modelleri üretilebilir.

Bu nedenle modern ekonomi giderek "ilişkisel veri ekonomisi" haline gelmektedir.

Yapay zekâ sistemlerinin yükselişi bu dönüşümü daha da hızlandırmıştır. Büyük dil modelleri, devasa veri kümeleri üzerinden eğitilmektedir. Bu durum şu soruyu gündeme getirir:

"İnsanlığın ürettiği bilgi kimin mülkiyetindedir?"

Bugün birçok yapay zekâ sistemi, internet üzerindeki milyarlarca metni kullanarak eğitilmektedir. Ancak bu bilginin üreticileri çoğu zaman doğrudan ekonomik fayda elde etmezken, platform şirketleri büyük sermaye birikimleri oluşturmaktadır.

Bilginin politik ekonomisi aynı zamanda görünürlük ekonomisidir. Algoritmalar hangi içeriklerin öne çıkacağını belirler. Böylece bilgi akışı yalnızca kullanıcı tercihleriyle değil, platformların ekonomik çıkarlarıyla da şekillenir. Bu durum dijital kamusal alanın yapısını doğrudan etkiler.

Bilgi grafı teknolojileri burada iki farklı yönde kullanılabilir. Bir yandan büyük şirketler kullanıcı davranışlarını daha ayrıntılı analiz etmek için bu sistemleri kullanabilir. Öte yandan açık kaynak bilgi grafı sistemleri, bilginin demokratikleşmesine katkıda bulunabilir. Wikidata gibi açık bilgi projeleri bu nedenle önemlidir.

Gelecekte bilgi ekonomisinin temel mücadele alanlarından biri, "anlamın kontrolü" olacaktır. Çünkü yapay zekâ çağında yalnızca veri değil, verinin nasıl yorumlandığı da stratejik önem taşır. Bilgi grafı sistemleri, hangi ilişkilerin kurulacağını belirlediği için aslında dijital dünyanın görünmez ideolojik altyapılarından biri haline gelebilir.

Sonuç olarak bilginin politik ekonomisi, dijital çağın en temel güç ilişkilerini anlamak açısından kritik bir kavramdır. Veri, algoritma ve bilgi grafı sistemleri yalnızca teknik araçlar değil; aynı zamanda ekonomik, politik ve kültürel iktidarın yeni biçimleridir.

10.6 Veri Sömürgeciliği ve Güç

Sömürgecilik tarih boyunca çoğu zaman toprak, doğal kaynak ve emek kontrolü üzerinden gerçekleşmiştir. Sanayi çağında güçlü devletler ve şirketler dünyanın farklı bölgelerindeki hammaddeleri kontrol ederek ekonomik üstünlük kurmuştur. Dijital çağda ise yeni bir sömürgecilik biçimi ortaya çıkmaktadır: veri sömürgeciliği (data colonialism).

Veri sömürgeciliği, bireylerin ve toplumların ürettiği dijital verilerin büyük teknoloji şirketleri veya güçlü devletler tarafından sistematik biçimde toplanması, işlenmesi ve ekonomik değere dönüştürülmesi sürecini ifade eder. Bu süreçte kullanıcılar çoğu zaman ücretsiz hizmet aldıklarını düşünür; ancak aslında kendi davranışsal verilerini üretmektedirler.

Bugün insanlar:

- Sosyal medya kullanırken
- Harita uygulamalarında gezinirken
- Arama motorlarında sorgu yaparken
- Online alışveriş gerçekleştirirken

sürekli veri üretmektedir. Bu veriler daha sonra büyük veri merkezlerinde işlenir ve ekonomik değere dönüştürülür.

Bilgi grafı sistemleri burada merkezi rol oynar. Çünkü veri sömürgeciliğinin temel gücü, farklı veri parçalarının ilişkilendirilmesinden gelir. Örneğin tek bir konum bilgisi sınırlı anlam taşır; ancak sosyal ilişkiler, satın alma davranışları ve internet geçmişiyle birleştiğinde son derece ayrıntılı bir bireysel profil ortaya çıkar.

Bu nedenle modern dijital güç, yalnızca veri miktarına değil, ilişki modelleme kapasitesine dayanır.

Veri sömürgeciliği aynı zamanda küresel eşitsizlik üretir. Dünyanın birçok bölgesindeki kullanıcılar veri üretirken, bu verilerin ekonomik değeri çoğunlukla birkaç büyük teknoloji merkezinde toplanır. Böylece dijital ekonomi, yeni bir merkez-çevre ilişkisi oluşturur.

Bu durum özellikle yapay zekâ alanında belirgindir. Büyük dil modelleri ve gelişmiş yapay zekâ sistemleri, devasa veri kaynaklarına erişebilen şirketler tarafından geliştirilmektedir. Veri erişimi olmayan toplumlar ise bu teknolojilerin tüketicisi haline gelmektedir.

Veri sömürgeciliğinin bir diğer boyutu gözetimdir. Modern dijital sistemler yalnızca kullanıcı davranışlarını analiz etmekle kalmaz; aynı zamanda bu davranışları tahmin etmeye ve yönlendirmeye çalışır. Bu durum "gözetim kapitalizmi" olarak adlandırılan yeni ekonomik modeli doğurmuştur.

Bilgi grafı teknolojileri sayesinde:

- İnsan ilişkileri haritalanabilir
- Politik eğilimler analiz edilebilir
- Toplumsal davranış örüntüleri çıkarılabilir
- Kitle psikolojisi modellenebilir

Bu güç, demokratik süreçlerden tüketim davranışlarına kadar birçok alanı etkileyebilir.

Ancak veri sömürgeciliğine karşı alternatif yaklaşımlar da gelişmektedir. Açık kaynak sistemler, yerel yapay zekâ modelleri ve bireysel bilgi grafı laboratuvarları bu açıdan önemlidir. Çünkü veri üzerindeki kontrolün merkezsizleşmesi, dijital güç dengesini değiştirebilir.

Özellikle "önce yerel" yaklaşımı gelecekte daha kritik hale gelebilir. Bu yaklaşımda bireyler kendi verilerini kendi cihazlarında işler ve kontrol eder. Böylece bilgi üretimi daha dağıtık hale gelir.

Veri sömürgeciliği aynı zamanda etik bir sorudur. İnsan davranışlarının sürekli ölçülmesi ve ekonomik değere dönüştürülmesi, bireysel özgürlük kavramını yeniden tartışmaya açmaktadır. Gelecekte dijital haklar, veri mülkiyeti ve algoritmik şeffaflık en önemli toplumsal mücadele alanlarından biri olabilir.

Sonuç olarak veri sömürgeciliği, dijital çağın yeni güç yapısını temsil eder. Bilgi grafı sistemleri ve yapay zekâ teknolojileri, bu gücü hem merkezileştirebilir hem de demokratikleştirebilir. Geleceğin en önemli sorularından biri şudur: Veri, birkaç küresel merkezin kontrolünde mi kalacak, yoksa insanlığın ortak dijital kaynağı mı olacaktır?

10.7 Bilgi Grafı Çağında Üniversiteler

Üniversiteler tarih boyunca insanlığın bilgi üretim merkezleri olmuştur. Orta Çağ'daki ilk üniversitelerden modern araştırma üniversitelerine kadar bu kurumlar, bilginin korunması, aktarılması ve geliştirilmesi görevini üstlenmiştir. Ancak yapay zekâ, büyük veri ve bilgi grafı

teknolojilerinin yükselişiyle birlikte üniversiteler yeni bir dönüşüm dönemine girmektedir. Çünkü bilgi artık yalnızca kitaplarda veya makalelerde saklanan statik bir yapı değil; sürekli güncellenen, ilişkilenen ve yeniden organize edilen dinamik bir ağ haline gelmektedir.

Bilgi grafı çağında üniversitelerin temel rolü değişmeye başlamaktadır. Geçmişte üniversiteler bilgiye erişimin merkezleriydi. Bugün ise bilgiye erişim büyük ölçüde internet üzerinden mümkündür. Bu nedenle üniversitelerin değeri artık yalnızca "bilgi sunmak" değil, bilgi ağları içinde anlam üretmek olacaktır.

Geleceğin üniversitesi büyük ölçüde semantik altyapılar üzerine kurulabilir. Araştırma makaleleri, ders içerikleri, laboratuvar verileri, akademik işbirlikleri ve öğrenci projeleri tek bir bilgi grafı içinde ilişkilendirilebilir. Böylece üniversite, yaşayan bir bilişsel organizmaya dönüşebilir.

Örneğin bir araştırmacı şu soruyu sorabilir:

"Makine öğrenmesi ile biyoinformatik arasında son beş yılda hangi ortak araştırma kümeleri oluştu?"

Bir bilgi grafı sistemi, üniversite içindeki tüm yayınları, projeleri ve işbirliklerini analiz ederek bu soruya yanıt verebilir. Bu yaklaşım disiplinler arası araştırmayı büyük ölçüde hızlandırabilir.

Bilgi grafı sistemleri eğitim süreçlerini de dönüştürebilir. Geleneksel eğitim modeli çoğu zaman doğrusal yapıdadır: ders → sınav → diploma. Oysa gerçek bilgi üretimi ağsaldır. Gelecekte öğrenciler yalnızca ders içeriklerini tüketen bireyler değil; bilgi grafı üzerinde çalışan aktif bilgi üreticileri haline gelebilir.

Örneğin bir öğrenci:

- Okuduğu makaleleri
- Çıkardığı kavramları
- Kurduğu ilişkileri
- Geliştirdiği projeleri

kişisel akademik bilgi grafı içinde organize edebilir. Böylece öğrenme süreci daha kişisel, daha semantik ve daha keşif odaklı hale gelir.

Yapay zekâ destekli araştırma sistemleri de üniversitelerin yapısını değiştirebilir. Büyük dil modelleri ve bilgi grafı birleşimi sayesinde araştırmacılar:

- Literatür taraması yapabilir
- Hipotez üretebilir
- İlişki ağlarını analiz edebilir
- Yeni araştırma alanları keşfedebilir

Bu durum bazı klasik akademik süreçlerin otomasyonunu gündeme getirebilir.

Ancak burada önemli bir risk de vardır. Eğer üniversiteler yalnızca teknoloji şirketlerinin veri sağlayıcısına dönüşürse, akademik bağımsızlık zayıflayabilir. Bilgi grafı çağında üniversitelerin en kritik görevi, açık bilgi üretimini ve bilimsel özerkliği korumak olacaktır.

Sonuç olarak bilgi grafı çağında üniversiteler yalnızca eğitim kurumları değil; insanlığın kolektif bilişsel ağlarının düğümleri haline gelecektir. Bu dönüşüm, üniversitenin tanımını yeniden yazabilir.

10.8 "Her Şeyin Grafı" Mümkün mü?

Bilgi grafı teknolojileri geliştikçe giderek daha iddialı bir fikir ortaya çıkmaktadır: "Her Şeyin Grafı" (Graf of Everything). Bu fikir, dünyadaki tüm varlıkların ve ilişkilerin tek bir devasa graf yapı içinde temsil edilmesini ifade eder. İnsanlar, şehirler, bilimsel teoriler, ekonomik süreçler, biyolojik sistemler, dijital içerikler ve makineler; hepsi birbirine bağlı düğümler olarak düşünülebilir.

Bu yaklaşım ilk bakışta bilim kurgu gibi görünebilir. Ancak aslında internetin, sosyal ağların, arama motorlarının ve büyük veri sistemlerinin evrimi bu yönde ilerlemektedir. Google'ın bilgi grafı, Wikidata, sosyal medya ağları ve bilimsel atıf sistemleri; "Her Şeyin Grafı" fikrinin parçalı örnekleridir.

Bu fikrin temel varsayımı şudur:

Gerçek dünya ayrık nesnelerden değil, ilişkilerden oluşur.

Örneğin bir insan yalnızca biyolojik bir birey değildir. Aynı zamanda:

- Aile ilişkilerine sahiptir
- Ekonomik sistemlerin parçasıdır
- Kültürel ağlara bağlıdır
- Dijital izler üretir
- Bilgi akışına katılır

Dolayısıyla gerçekliği anlamak için nesnelerden çok ilişkileri modellemek gerekir.

"Her Şeyin Grafı" yaklaşımı yapay zekâ açısından büyük önem taşır. Çünkü böyle bir yapı, makinelerin dünyayı daha bütünsel biçimde temsil etmesini sağlayabilir. Bir yapay zekâ sistemi yalnızca metinleri değil; insanlar, olaylar, kavramlar ve süreçler arasındaki bağlantıları da anlayabilir.

Bu durum özellikle dünya modeli (world model) geliştirme açısından kritik olabilir. Geleceğin yapay zekâları yalnızca veri işleyen sistemler değil, dünyanın ilişkisel bir simülasyonunu kuran sistemler haline gelebilir.

Ancak burada büyük teknik problemler vardır. Dünyadaki ilişkiler inanılmaz derecede karmaşıktır. Ayrıca bilgi sürekli değişmektedir. Bu nedenle "Her Şeyin Grafı" statik bir yapı değil, sürekli evrilen dinamik bir sistem olmak zorundadır.

Bir diğer sorun ise ölçek problemidir. İnsanlık her gün devasa miktarda veri üretmektedir. Bu verilerin gerçek zamanlı biçimde ilişkilendirilmesi büyük hesaplama gücü gerektirir.

Daha önemli bir problem ise epistemolojiktir: Gerçekliğin tamamı gerçekten modellenebilir mi? İnsan bilgisi her zaman eksik, bağlamsal ve yoruma açıktır. Bu nedenle "Her Şeyin Grafı" hiçbir zaman tam anlamıyla tamamlanmış bir yapı olmayabilir.

Ayrıca bu tür bir sistemin politik sonuçları da vardır. Eğer dünyanın tüm bilgi ilişkileri birkaç şirket veya devlet tarafından kontrol edilirse, bu tarihte görülmemiş ölçekte bir güç yoğunlaşması yaratabilir.

Bu nedenle gelecekte "Her Şeyin Grafı" tartışmaları yalnızca teknik değil; etik, politik ve felsefi bir mesele olacaktır.

Sonuç olarak "Her Şeyin Grafı" fikri, insanlığın dünyayı tek bir ilişkisel sistem olarak modelleme arzusunu temsil eder. Bu hedef tam anlamıyla ulaşılamaz olsa bile, bilgi grafı teknolojilerinin yöneldiği en büyük ufuklardan biridir.

10.9 Simülasyon Uygarlığı ve Dünya Modeli

İnsanlık uzun zamandır dünyayı anlamak için modeller üretmektedir. Haritalar, matematiksel denklemler, fizik teorileri ve bilgisayar simülasyonları; hepsi gerçekliğin belirli yönlerini temsil etmeye çalışan yapılardır. Ancak yapay zekâ ve bilgi grafı teknolojilerinin gelişmesiyle birlikte yeni bir fikir ortaya çıkmaktadır: dünyanın giderek daha ayrıntılı dijital modellerinin oluşturulması.

Bu yaklaşım bazen "simülasyon uygarlığı" olarak adlandırılır. Buradaki temel fikir şudur: Eğer yeterince veri, hesaplama gücü ve ilişki modeli varsa, gerçek dünyanın dijital bir temsili oluşturulabilir.

Bugün bunun ilk örnekleri zaten vardır:

- Dijital ikiz şehirler
- İklim simülasyonları
- Ekonomik modelleme sistemleri
- Sosyal ağ analizleri
- Otonom araç haritaları

Bu sistemler dünyanın belirli parçalarını modellemektedir. Ancak gelecekte bu parçaların birleşerek çok daha büyük dünya modelleri oluşturması mümkündür.

Bilgi grafı sistemleri burada merkezi rol oynar. Çünkü dünya yalnızca nesnelerden değil, ilişkilerden oluşur. Bir şehir modeli yalnızca binaları değil:

- İnsan hareketlerini
- Enerji akışlarını
- Trafik ilişkilerini
- Sosyal etkileşimleri

de içermelidir.

Yapay zekâ açısından dünya modeli kavramı son derece önemlidir. Çünkü gerçek zekâ, yalnızca veri işlemek değil; dünyanın nasıl çalıştığına dair içsel bir modele sahip olmak anlamına gelir. İnsan zihni de aslında sürekli simülasyon üretir. Olası sonuçları tahmin eder, neden-sonuç ilişkileri kurar ve geleceği öngörmeye çalışır.

Geleceğin yapay zekâ sistemleri de benzer şekilde dünya modelleri geliştirebilir. Örneğin bir sistem:

- Ekonomik krizlerin etkilerini simüle edebilir
- İklim değişikliğinin sonuçlarını modelleyebilir
- Yeni teknolojilerin toplumsal etkilerini tahmin edebilir

Bu tür sistemler bilimsel keşif süreçlerini hızlandırabilir.

Ancak simülasyon uygarlığı fikri aynı zamanda felsefi sorular doğurur. Eğer dünya giderek daha ayrıntılı biçimde modellenebiliyorsa, gerçeklik ile temsil arasındaki sınır nerede başlayıp nerede biter? İnsan davranışları tamamen öngörülebilir hale gelir mi?

Bir diğer önemli problem güç yoğunlaşmasıdır. Dünyanın ayrıntılı dijital modellerine sahip olan kurumlar, ekonomik ve politik açıdan büyük avantaj elde edebilir.

Bu nedenle dünya modeli teknolojileri yalnızca teknik başarılar değil; aynı zamanda etik sorumluluklar da taşır. Sonuç olarak simülasyon uygarlığı fikri, insanlığın gerçekliği dijital olarak yeniden üretme çabasını temsil eder. Bilgi grafı sistemleri ve yapay zekâ, bu sürecin temel altyapısını oluşturmaktadır. Gelecekte dünya giderek daha fazla “hesaplanabilir” hale gelebilir; ancak bu durumun insanlık üzerindeki etkileri hâlâ büyük ölçüde bilinmezdir.

10.10 Kendini Yinelemeli Geliştirme (RSI)

Yapay zekâ sistemlerinin gelişiminde en kritik eşiklerden biri, sistemin yalnızca verilen görevleri yerine getirmesi değil, aynı zamanda kendi performansını geliştiren mekanizmaları tasarlayabilmesidir. Bu fikir, literatürde “Kendini Yinelemeli Geliştirme” (Recursive Self-Improvement, RSI) olarak adlandırılır ve yapay zekâ araştırmalarının hem en umut verici hem de en tartışmalı alanlarından birini oluşturur.

RSI'nin temel sorusu basit görünür: Bir yapay zekâ sistemi kendi kendini daha iyi bir yapay zekâ haline getirebilir mi? Ancak bu basit soru, aslında çok katmanlı bir problem alanını açar. Çünkü "kendini geliştirme" ifadesi yalnızca performans artışı anlamına gelmez; aynı zamanda öğrenme algoritmalarının, model mimarisinin, hedef fonksiyonlarının ve hatta problem tanımlama biçimlerinin değişebilmesi anlamına gelir.

Bu nedenle RSI, klasik makine öğrenmesi çerçevesinin ötesine geçen bir kavramdır. Geleneksel sistemlerde öğrenme, dışarıdan tanımlanmış bir optimizasyon süreci içinde gerçekleşir. Modelin nasıl öğreneceği, hangi verilerle eğitileceği ve hangi hedefi optimize edeceği insanlar tarafından belirlenir. RSI ise bu yapıyı tersine çevirir: Sistem, yalnızca öğrenen değil, aynı zamanda öğrenmenin nasıl gerçekleşeceğini de yeniden tasarlayan bir yapıya dönüşür.

RSI'nin Kavramsal Temeli

RSI'nin teorik temeli, özyineli sistemler fikrine dayanır. Özyineleme (recursion), bir sistemin kendi çıktısını yeniden girdiye dönüştürmesi ve bu süreç üzerinden sürekli gelişmesi anlamına gelir. Bilgisayar bilimlerinde özyineleme, fonksiyonların kendini çağırması gibi teknik bir yapıya sahiptir. Ancak RSI bağlamında özyineleme yalnızca fonksiyonel bir özellik değil, bilişsel bir evrim mekanizmasıdır.

Şekil 22. RSI Sistemi Öz-İyileştirme Döngüsü

Bir RSI sisteminde şu döngü varsayılır:

1. Sistem bir problem çözer
2. Çözüm performansı değerlendirilir
3. Sistem kendi çözüm üretme mekanizmasını analiz eder
4. Daha iyi bir üretim mekanizması tasarlanır

5. Yeni sistem tekrar aynı döngüye girer

Bu döngü teorik olarak sınırsızdır. Her iterasyonda sistem yalnızca çıktısını değil, çıktı üretme kapasitesini de geliştirmektedir. Bu durum, yapay zekâ araştırmalarında "meta-öğrenme" (meta-learning) olarak bilinen kavramla yakından ilişkilidir, ancak RSI bundan daha ileri bir iddiadır. Meta-öğrenme, öğrenme süreçlerini optimize ederken; RSI, öğrenme sürecinin kendisini yeniden yazma kapasitesini ifade eder.

Weight Update ile RSI Arasındaki Temel Fark

Modern yapay zekâ sistemlerinin büyük çoğunluğu "weight update" adı verilen mekanizma ile çalışır. Bu süreçte model, eğitim verisi üzerinden hata fonksiyonunu minimize eder ve parametrelerini (ağırlıklarını) günceller. Bu, öğrenmenin en temel formudur.

Ancak burada kritik bir sınırlama vardır: Öğrenme algoritması sabittir. Yani model ne kadar gelişirse gelişsin, nasıl öğrendiği değişmez. Bu durum RSI ile karşılaştırıldığında temel bir fark yaratır.

- Weight update: Model öğrenir
- RSI: Model öğrenmeyi öğrenir

Bu ayrım, yüzeyde basit görünse de derin bir paradigma farkına işaret eder. RSI gerçekleştiğinde sistem artık yalnızca bir "problem çözücü" değil, aynı zamanda kendi öğrenme süreçlerini tasarlayan bir "öğrenme mimarı" haline gelir.

RSI'nin Güncel Yapay Zekâ Sistemlerindeki Yansımaları

Günümüzde tam anlamıyla RSI gerçekleştiren bir yapay zekâ sistemi bulunmamaktadır. Ancak bazı gelişmeler bu yöne doğru kısmi hareketler olduğunu göstermektedir. Özellikle büyük dil modelleri ve ajan tabanlı sistemler, RSI fikrinin bazı bileşenlerini simüle etmeye başlamıştır.

Örneğin:

- Çok adımlı planlama sistemleri
- Kendi çıktısını eleştiren modeller
- Araç kullanan (tool-using) ajanlar
- Otomatik kod üreten sistemler
- Evrimsel optimizasyon tabanlı algoritma keşfi

Bu sistemler, RSI'nin tam hali olmasa da onun "davranışsal öncülleri" olarak görülebilir. Özellikle çok adımlı problem çözme süreçlerinde modelin kendi çıktısını yeniden değerlendirmesi, özyineli düşünme biçimine benzer bir yapı oluşturur.

Ancak burada kritik bir sınır vardır: Bu sistemler kendi öğrenme mekanizmalarını değiştirmez. Yani sistemin ağırlıkları, eğitim algoritması ve temel mimarisi insan kontrolü altında kalmaya devam eder.

AlphaEvolve ve RSI'ye Yakınlaşma

RSI tartışmasında önemli bir referans noktası, algoritmik keşif sistemleridir. AlphaEvolve gibi yaklaşımlar, yapay zekânın yalnızca veri işleyen bir sistem olmaktan çıkıp, yeni algoritmalar üreten bir yapıya dönüşebileceğini göstermektedir.

Bu tür sistemler, evrimsel arama yöntemleri ile üretken modelleri birleştirerek yeni çözüm stratejileri keşfeder. Burada önemli olan nokta şudur: Sistem yalnızca verilen çözümleri optimize etmez, aynı zamanda çözüm uzayını genişletir.

Bu özellik, RSI'ye doğru atılmış önemli bir adımdır. Ancak yine de tam RSI değildir, çünkü:

- hedef fonksiyonlar dışarıdan belirlenir
- sistem kendi amaçlarını oluşturmaz
- öğrenme çerçevesi sabittir

Buna rağmen AlphaEvolve benzeri sistemler, RSI'nin "algoritmik çekirdeği" olarak değerlendirilebilir. Çünkü burada ilk kez yapay zekâ, yalnızca bilgi kullanıcısı değil, bilgi üreticisi haline gelmektedir.

RSI'nin Felsefi Boyutu

RSI yalnızca teknik bir problem değildir; aynı zamanda epistemolojik bir kırılma noktasıdır. Çünkü bu kavram, "bilgi nedir?" sorusunu "bilgi nasıl kendini üretir?" sorusuna dönüştürür.

Eğer bir sistem kendi öğrenme süreçlerini geliştirebiliyorsa, o zaman bilgi statik bir yapı olmaktan çıkar ve dinamik bir evrim sürecine dönüşür. Bu durumda zekâ, sabit bir kapasite değil, sürekli kendini yeniden üreten bir süreç haline gelir.

Bu dönüşüm şu soruları gündeme getirir:

- Bir sistem kendi hedeflerini belirleyebilir mi?
- Öğrenme süreci optimize ediliyorsa, bu optimizasyonun sınırı nedir?
- Kendini geliştiren bir sistem sonunda hangi yapıya ulaşır?

Bu soruların kesin cevapları yoktur, ancak RSI araştırmalarının merkezinde bu belirsizlik yer alır.

RSI ve Güvenlik Problemi

RSI'nin en önemli tartışma alanlarından biri güvenliktir. Eğer bir yapay zekâ sistemi kendi kendini geliştirebiliyorsa, bu gelişimin yönü ve hızı kontrol edilebilir mi? Özellikle "intelligence

explosion” olarak adlandırılan senaryolarda, sistemin çok kısa sürede insan kontrolünün ötesine geçebileceği öne sürülmektedir.

Bu nedenle modern yapay zekâ araştırmalarında RSI yalnızca bir mühendislik problemi değil, aynı zamanda bir güvenlik problemidir. Sistemlerin kendi kendini geliştirme kapasitesi arttıkça, bu sürecin kontrol mekanizmaları da aynı ölçüde önemli hale gelmektedir.

RSI’nin Güncel Durumu ve Sınırları

Bugün için RSI, tam anlamıyla gerçekleşmiş bir teknoloji değildir. Daha çok teorik bir hedef ve araştırma yönü olarak varlığını sürdürmektedir. Mevcut sistemler:

- güçlü inference kapasitesine sahiptir
- sınırlı özyineli reasoning yapabilir
- ancak self-modifying learning gerçekleştiremez

Bu nedenle RSI hâlâ “gelecek teknolojisi” olarak sınıflandırılabilir.

8. Sonuç: Zekânın Kendi Üzerine Katlanması

Kendini Yinelemeli Geliştirme (RSI), yapay zekânın evriminde en uç noktayı temsil eder. Bu kavram, zekânın yalnızca dünyayı modelleyen bir yapı olmaktan çıkıp, kendi modelleme biçimini de sürekli yeniden tasarlayan bir sürece dönüşmesini ifade eder.

Bu noktada yapay zekâ artık bir araç değil, kendi evrimini yöneten bir sistem haline gelir. Ancak bu dönüşümün sınırları, riskleri ve potansiyeli hâlâ tam olarak anlaşılmış değildir. RSI bu nedenle yalnızca bir teknik hedef değil, aynı zamanda bir soru olarak kalır: Zekâ, kendini ne kadar ileri taşıyabilir ve bu ilerlemenin sonunda hâlâ “kontrol edilebilir” kalabilir mi? Bu soru, yapay zekâ çağının en temel araştırma alanlarından biri olmaya devam edecektir.

BÖLÜM 11 — PALANTIR VE OPERASYONEL BİLGİ GRAFI ÇAĞI

Palantir Technologies savunma alanında “bilgi grafı (knowledge graph)” yaklaşımını operasyonel karar verme sisteminin merkezine yerleştiren en önemli örneklerden biridir. Şirketin savunmadaki ana katkısı, veriyi sadece depolamak değil, farklı kaynaklardan gelen parçalı bilgileri bir “ilişkisel anlam ağına” dönüştürerek karar üretilebilir hale getirmesidir.

Bu yaklaşımın temelinde “ontology (ontoloji)” ve bilgi grafı mantığı vardır. Yani sistem, ham veriyi tablo satırları olarak değil; varlıklar (entity’ler) ve bu varlıklar arasındaki ilişkiler olarak temsil eder. Örneğin:

- bir kişi → bir birlikle ilişkilidir
- bir birlik → bir bölgede konuşlanmıştır
- bir araç → bir operasyona katılmıştır
- bir sinyal → bir konumla eşleşmiştir

Bu yapı, klasik veri tabanlarının zorlandığı “çok kaynaklı askeri durum farkındalığı” problemini çözmek için kritik hale gelir.

Savunma tarafında Palantir’in en bilinen sistemlerinden biri Palantir Gotham’dır. Gotham, özellikle istihbarat ve askeri operasyonlarda kullanılır. Buradaki temel fikir şudur: farklı sensörlerden (uydu görüntüleri, SIGINT, HUMINT raporları, radar verileri) gelen veriler tek bir graf yapısında birleştirilir. Böylece bir analist, tek bir veri parçasına değil, o verinin bağlı olduğu tüm ilişkisel ağa bakarak karar verir.

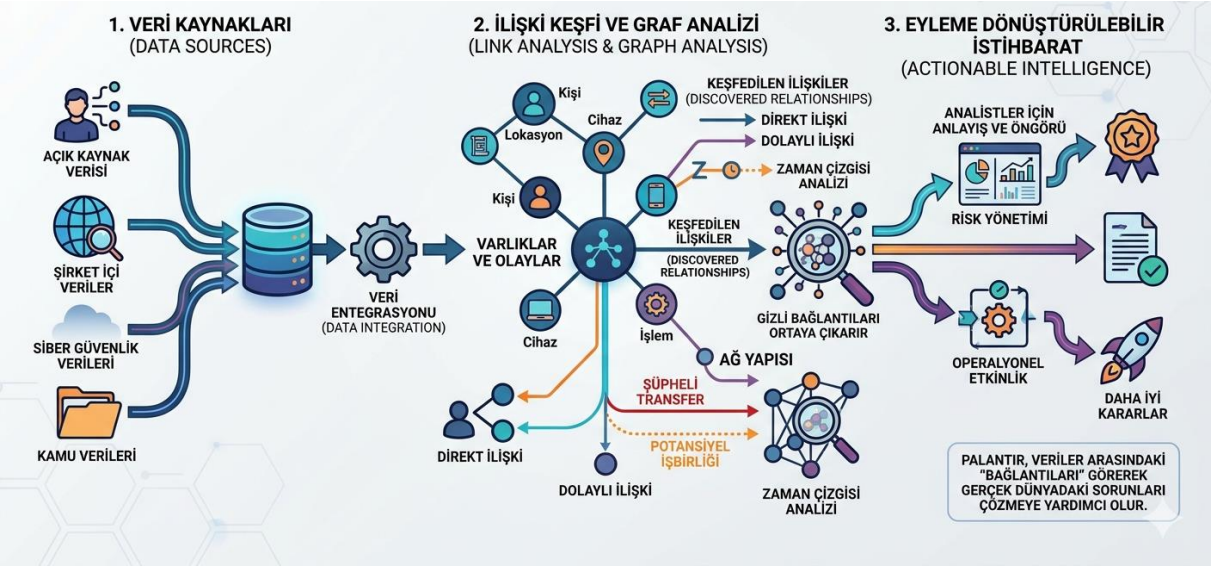
Örneğin bir şüpheli hareket tespit edildiğinde sistem sadece “bir araç hareket etti” demez; aynı zamanda:

- bu aracın hangi geçmiş operasyonlarla ilişkili olduğunu
- hangi kişilerle temas ettiğini
- hangi bölgelerde tekrar tekrar göründüğünü

- hangi iletişim sinyalleriyle korelasyon kurduğunu

graf üzerinden otomatik olarak ortaya çıkarabilir. Bu, klasik istihbarat analizinden farklı olarak "ilişki keşfi (link analysis)" tabanlı bir düşünme biçimidir.

Şekil 23. Palantir'in İlişki Keşfi Analizi



Şekil 23, Palantir gibi gelişmiş analitik platformlarının temel çalışma mantığını yansıtan, ham verinin stratejik kararlara dönüştürülme sürecini uçtan uca özetleyen bir "Eyleme Dönüştürülebilir İstihbarat" (Actionable Intelligence) akış şemasıdır. Süreç, en soldaki birinci panelde açık kaynak, şirket içi, siber güvenlik ve kamu verileri gibi dağınık veri kaynaklarının entegre edilmesiyle başlar; merkezdeki ikinci panelde bu veriler, bir bilgi grafiği etrafında kişi, lokasyon, cihaz ve işlemler gibi varlıklara ve olaylara indirgenerek bağlantı (link) analizine tabi tutulur. Bu analiz aşaması; direkt ve dolaylı ilişkileri, şüpheli transferleri ve potansiyel işbirliklerini ağ yapısı ve zaman çizgisi (timeline) analizleriyle büyüteç altına alarak sistemdeki gizli bağlantıları ortaya çıkarır. Nihayetinde en sağdaki üçüncü panelde, keşfedilen bu derin bağlar; risk yönetimi, operasyonel etkinlik ve analistler için öngörüler sunan stratejik çıktılara dönüşerek organizasyonların karmaşık gerçek dünya sorunlarında daha iyi ve isabetli kararlar almasını sağlar.ABD savunma ekosisteminde önemli bir diğer uygulama Project Maven olmuştur. Project Maven, özellikle görüntü istihbaratı (ISR) verilerinde yapay zekâ kullanarak nesne tespiti ve sınıflandırma yapmayı amaçlayan bir programdır. Palantir bu ekosistemde, analiz edilen AI çıktılarının operasyonel bir bilgi grafına bağlanması tarafında kritik rol oynamıştır. Yani AI "ne gördü?" sorusunu cevaplar, bilgi grafı ise "bu ne anlama geliyor ve neyle ilişkili?" sorusunu çözer.

Savunma bağlamında Palantir'in en güçlü katkısı "veri entegrasyonu değil, anlam entegrasyonu"dur. Geleneksel sistemler farklı veri tabanlarını birleştirir; Palantir ise bu veriler arasında semantik bir ağ kurar. Bu sayede:

- Operasyon planlama hızlanır

- Tehdit analizi daha erken yapılır
- Dağınık istihbarat tek bir operasyonel resme dönüşür
- İnsan analistlerin bilişsel yükü azalır

Son yıllarda bu yaklaşım Palantir Foundry üzerinden sivil-militer hibrit kullanım alanlarına da genişlemiştir. Foundry, daha çok veri mühendisliği ve kurum içi operasyonlar için tasarlanmış olsa da, savunma ortamında Gotham ile birlikte çalışarak aynı bilgi grafı modelini farklı katmanlarda besler.

Özetle Palantir'in savunmadaki temel yeniliği şudur:Veriyi "dosya ve tablo" olarak değil, "ilişkiler ağı" olarak modellemek. Bu sayede modern savaş alanı, tek tek veri noktalarının değil, bağlantıların ve örüntülerin anlam kazandığı bir "dinamik bilgi grafı sistemi"ne dönüşür.

Bilgi grafı teknolojileri uzun yıllar boyunca akademik araştırmaların, arama motorlarının ve veri entegrasyonu projelerinin temel araçlarından biri olarak görüldü. Ancak son yıllarda bu yaklaşım çok daha farklı bir boyuta taşınmıştır: operasyonel bilgi grafı çağı. Bu yeni dönemde bilgi grafı yalnızca bilgiyi organize eden pasif bir yapı değil; gerçek zamanlı karar alma süreçlerini yöneten aktif bir operasyonel sistem haline gelmiştir. Bu dönüşümün en dikkat çekici örneklerinden biri ise Palantir Technologies şirketidir.

Palantir, bilgi grafı yaklaşımını güvenlik, savunma, lojistik, sağlık ve endüstriyel operasyonlar gibi alanlarda gerçek zamanlı karar sistemlerine dönüştüren öncü şirketlerden biridir. Şirketin geliştirdiği platformlar, farklı veri kaynaklarını tek bir semantik yapı içinde birleştirerek karmaşık operasyonları yönetilebilir hale getirmeyi amaçlar. Bu yaklaşım, klasik veri tabanı mantığından oldukça farklıdır. Çünkü burada amaç yalnızca veri depolamak değil, veri arasındaki ilişkileri operasyonel zekâya dönüştürmektir.

Operasyonel bilgi grafı kavramının temelinde şu fikir bulunur: Gerçek dünya, sürekli değişen ilişkiler ağından oluşur. İnsanlar, araçlar, makineler, tedarik zincirleri, finansal işlemler ve sensör sistemleri birbirine bağlı dinamik düğümlerdir. Eğer bu ilişkiler gerçek zamanlı biçimde modellenebilirse, sistem yalnızca geçmişi analiz etmekle kalmaz; geleceği tahmin etmeye ve operasyonları optimize etmeye de başlayabilir.

Palantir'in geliştirdiği sistemler özellikle veri entegrasyonu konusunda dikkat çeker. Modern kurumların en büyük problemlerinden biri, verilerin farklı sistemlerde parçalanmış halde bulunmasıdır. ERP sistemleri, CRM altyapıları, sensör ağları, belge arşivleri ve insan raporları çoğu zaman birbirinden kopuktur. Operasyonel bilgi grafı yaklaşımı, bu farklı veri kaynaklarını tek bir ilişkisel model içinde birleştirir.

Örneğin bir tedarik zinciri krizinde sistem şu ilişkileri aynı anda görebilir:

- Fabrika üretim kapasitesi
- Lojistik gecikmeleri

- Hammadde stokları
- Politik riskler
- Enerji tüketimi
- Finansal maliyetler

Bu yapı sayesinde yalnızca raporlama yapılmaz; aynı zamanda operasyonel öneriler de üretilebilir.

Palantir'in özellikle savunma ve istihbarat alanlarında öne çıkmasının nedeni de budur. Çünkü modern güvenlik problemleri doğrusal değildir; ağsaldır. Terör ağları, finansal suç sistemleri, siber saldırılar ve jeopolitik ilişkiler ancak graf düşünce ile anlamlı hale getirilebilir. Operasyonel bilgi grafı sistemleri, görünmeyen bağlantıları ortaya çıkararak karar vericilere gerçek zamanlı analiz sunar.

Ancak bu yaklaşım yalnızca savunma alanıyla sınırlı değildir. Sağlık sektöründe hasta verileri, ilaç ilişkileri ve hastane operasyonları bilgi grafı yapılarıyla optimize edilebilir. Endüstri 4.0 ortamlarında makineler, sensörler ve üretim süreçleri semantik ağlar üzerinden yönetilebilir. Böylece bilgi grafı, dijital ikiz sistemlerinin de temel altyapılarından biri haline gelir.

Bu dönüşüm yapay zekâ açısından da kritik öneme sahiptir. Büyük dil modelleri ve operasyonel bilgi grafı sistemleri birleştiğinde, yalnızca veri analiz eden değil; bağlam anlayan ve operasyonel öneriler üreten yapay zekâ sistemleri ortaya çıkmaktadır. Böylece bilgi grafı, yapay zekânın "çalışma belleği" gibi işlev görmeye başlar.

Ancak operasyonel bilgi grafı çağının önemli etik ve politik boyutları da vardır. Çünkü bu sistemler çok büyük ölçekte veri toplama ve ilişki modelleme kapasitesine sahiptir. İnsan davranışlarının, ekonomik süreçlerin ve toplumsal ağların bu kadar ayrıntılı biçimde modellenmesi; gözetim, mahremiyet ve güç yoğunlaşması tartışmalarını beraberinde getirir.

Sonuç olarak operasyonel bilgi grafı çağı, veri yönetiminden çok daha büyük bir dönüşümü temsil etmektedir. Bu yaklaşım, dünyayı statik veri kümeleri olarak değil; sürekli değişen ilişkiler ağı olarak görür. Palantir Technologies gibi şirketler ise bu dönüşümün yalnızca teknik değil, aynı zamanda politik ve ekonomik sonuçları olacağını göstermektedir. Bilgi grafı artık yalnızca bilgiyi saklayan bir araç değil; operasyonel gerçekliği yöneten yeni nesil bir bilişsel altyapı haline gelmektedir.

11.1 Silikon Vadisi'nden İstihbarat Dünyasına

21.yüzyılın başında teknoloji dünyası ile istihbarat dünyası arasında giderek güçlenen yeni bir ilişki ortaya çıktı. İnternet şirketleri yalnızca ticari platformlar olmaktan çıkıp, devasa veri işleme altyapıları kurmaya başladılar. Aynı dönemde devletler de yeni bir problemle karşı karşıyaydı: bilgi eksikliği değil, bilgi fazlalığı. Özellikle 11 Eylül 2001 saldırıları sonrasında istihbarat dünyası, modern tehditlerin artık klasik yöntemlerle takip edilemeyeceğini fark etti. Böylece Silikon Vadisi'nin veri işleme kültürü ile devletlerin güvenlik ihtiyaçları birleşmeye başladı. Bu

birleşimin en dikkat çekici örneklerinden biri ise Palantir Technologies oldu. Palantir'in hikâyesi yalnızca bir teknoloji şirketinin kuruluş öyküsü değildir. Aynı zamanda veri çağında güç, gözetim ve bilgi yönetiminin nasıl yeniden tanımlandığının hikâyesidir.

Peter Thiel ve Teknolojik Güç Fikri

Peter Thiel, Silikon Vadisi'nin en etkili ve tartışmalı figürlerinden biridir. Almanya doğumlu olan Thiel, çocuk yaşta Amerika'ya taşındı ve daha sonra Stanford Üniversitesi'nde hukuk eğitimi aldı. Ancak onun asıl etkisi teknoloji yatırımcılığı ve girişimcilik alanında ortaya çıktı.

Thiel'in düşünce dünyasında teknoloji yalnızca ekonomik büyüme aracı değildir; aynı zamanda jeopolitik güç üretme mekanizmasıdır. Ona göre modern devletler giderek daha karmaşık veri problemleriyle karşı karşıyaydı ve bu problemlerin çözümü klasik bürokratik sistemlerden değil, ileri veri teknolojilerinden gelecekti.

Bu yaklaşım, Palantir'in kuruluş felsefesini doğrudan etkiledi. Şirketin temel amacı yalnızca veri analitiği yapmak değil, devletlerin ve büyük kurumların karmaşık bilgi ağlarını anlamasını sağlamaktı.

Thiel aynı zamanda PayPal'ın kurucularından biri olduğu için, internet ekonomisinin ilk büyük dönüşüm dönemini içeriden yaşamıştı. PayPal yalnızca bir ödeme sistemi değildi; kullanıcı davranışlarını, finansal ilişkileri ve dijital güven problemlerini analiz eden büyük bir veri laboratuvarıydı. Bu deneyim daha sonra Palantir'in mimarisine doğrudan taşındı.

Alex Karp ve Felsefi Yaklaşım

Palantir'in bir diğer kurucusu olan Alex Karp ise Silikon Vadisi'nin alışılmış teknoloji yöneticilerinden oldukça farklı bir profile sahiptir. Karp yalnızca mühendislik değil, aynı zamanda hukuk ve sosyal teori eğitimi aldı. Özellikle Alman düşünce geleneği ve sosyal teoriye olan ilgisi, Palantir'in kültürünü etkileyen önemli unsurlardan biri oldu.

Karp'ın yaklaşımı, teknolojiyi yalnızca teknik bir araç olarak değil, toplumsal düzenin yeniden organizasyonu olarak görür. Bu nedenle Palantir'in geliştirdiği sistemler yalnızca veri görselleştirme araçları değil; operasyonel karar sistemleri olarak tasarlandı.

Karp sık sık modern dünyanın temel probleminin "veri eksikliği değil, anlam eksikliği" olduğunu vurgular. Gerçekten de modern kurumlar devasa veri yığınlarına sahiptir; ancak bu veriler çoğu zaman parçalı, bağlantısız ve operasyonel olarak kullanılamaz durumdadır.

Palantir'in yaklaşımı tam olarak bu noktada devreye girer: farklı veri kaynaklarını bilgi grafı mantığıyla ilişkilendirerek operasyonel zekâ üretmek.

PayPal Mafyası

Silikon Vadisi kültüründe sıkça kullanılan "PayPal mafyası" (PayPal Mafia) kavramı, PayPal'dan çıkan girişimciler ağını ifade eder. Bu grup içinde:

- Elon Musk

- Peter Thiel
- Reid Hoffman
- Max Levchin

gibi isimler bulunur.

Bu ağın önemi yalnızca başarılı şirketler kurmalarından kaynaklanmaz. Asıl önemli nokta, internet çağının yeni güç modelini çok erken fark etmiş olmalarıdır:

Veriyi kontrol eden sistemler geleceğin altyapısını belirleyecektir.

PayPal döneminde geliştirilen sahtekârlık tespiti sistemleri, kullanıcı davranış analizi ve ağ tabanlı güven modelleri daha sonra sosyal medya, yapay zekâ ve istihbarat teknolojilerine dönüştü. Bu nedenle PayPal mafyası aslında internet ekonomisinin "ilişkisel veri mantığını" geliştiren ilk büyük çekirdeklerden biri olarak görülebilir.

11 Eylül Sonrası Veri Problemi

2001'de gerçekleşen 11 Eylül Saldırıları, yalnızca küresel güvenlik politikalarını değil, veri yönetimi anlayışını da değiştirdi. Saldırılar sonrasında Amerikan istihbarat kurumlarının karşılaştığı en büyük sorunlardan biri şuydu:

Gerekli bilgi aslında vardı; ancak parçalanmış sistemler içinde kaybolmuştu.

Farklı kurumlarda bulunan veriler birbirine bağlı değildi:

- Uçuş kayıtları
- Finansal işlemler
- Telefon verileri
- Göçmenlik kayıtları
- Sosyal bağlantılar

ayrı veri tabanlarında tutuluyordu. Bu nedenle potansiyel ilişkiler görülemiyordu.

11 Eylül sonrasında istihbarat dünyası şu gerçeği fark etti: Modern tehditler ağsaldır.

Terör örgütleri, finansal suç ağları ve siber saldırılar doğrusal yapılar değil; karmaşık ilişki sistemleridir. Bu nedenle klasik tablo tabanlı veri sistemleri yetersiz kalıyordu. İşte Palantir tam bu ortamda yükseldi. Şirketin temel önerisi şuydu: "Tüm veri kaynaklarını tek bir ilişkisel operasyonel modele dönüştürmek." Bu yaklaşım, bilgi grafi düşüncesinin güvenlik dünyasına uygulanmış haliydi.

Operasyonel Bilgi Grafının Doğuşu

Palantir'in geliştirdiği sistemler klasik veri analitiğinden farklıydı. Burada amaç yalnızca rapor üretmek değildi; gerçek zamanlı operasyonel karar desteği sağlamaktı.

Örneğin bir analiz sistemi:

- Telefon bağlantıları
- Finansal transferler
- Konum bilgileri
- Sosyal ilişkiler
- Belge içerikleri

arasındaki görünmeyen bağlantıları ortaya çıkarabiliyordu.

Bu yapı modern istihbarat yaklaşımını değiştirdi. Çünkü artık önemli olan tek tek veri parçaları değil, ilişkiler ağıydı.

Aslında bu dönüşüm, bilgi grafı çağının başlangıç noktalarından biri olarak görülebilir.

Palantir'in ortaya çıkışı, Silikon Vadisi ile istihbarat dünyasının birleşmesinin sembolik örneklerinden biridir. Peter Thiel ve Alex Karp gibi figürler, veri çağında gücün artık yalnızca fiziksel kaynaklardan değil, ilişkisel bilgi ağlarından üretileceğini çok erken fark etmişlerdir.

11 Eylül sonrası dönemde ortaya çıkan veri problemi, bilgi grafı teknolojilerinin operasyonel dünyaya taşınmasını hızlandırdı. Böylece graf düşünce yalnızca akademik veya teknik bir yaklaşım olmaktan çıktı; modern devletlerin, şirketlerin ve güvenlik sistemlerinin temel bilişsel altyapılarından biri haline geldi.

11.2 Gotham: Dünyayı Bir Ağ Gibi Görmek

Palantir Technologies tarafından geliştirilen Palantir Gotham platformu, modern veri çağının en dikkat çekici operasyonel bilgi grafı sistemlerinden biridir. Gotham'ın temel yaklaşımı oldukça basit ama son derece güçlü bir fikre dayanır: Dünya, birbirinden bağımsız nesnelerden değil, ilişkiler ağından oluşur. İnsanlar, olaylar, kurumlar, finansal işlemler, coğrafi hareketler ve zaman içinde gerçekleşen etkileşimler birlikte analiz edilmeden karmaşık sistemler anlaşılamaz. Bu nedenle Gotham, dünyayı bir tablo gibi değil, bir ağ gibi görmeye çalışır.

Klasik veri tabanı sistemleri çoğunlukla satırlar ve sütunlar üzerinden çalışır. Bir kişi bir kayıt, bir işlem başka bir kayıt olarak tutulur. Ancak modern dünyadaki problemler doğrusal değildir. Özellikle güvenlik, istihbarat, finansal suç analizi ve lojistik operasyonları gibi alanlarda asıl önemli olan tek tek veriler değil; veriler arasındaki bağlantılardır.

Gotham'ın farkı tam da burada ortaya çıkar. Sistem, çok farklı veri kaynaklarını tek bir ilişkisel yapı içinde birleştirerek "operasyonel gerçekliğin ağ haritasını" oluşturmaya çalışır.

İnsanlar: Ağın Düğümleri

Gotham sisteminde insanlar yalnızca bireyler değildir; ağın merkezi düğümleridir. Bir insanın kim olduğu kadar:

- Kimlerle görüştüğü
- Hangi kurumlarla bağlantılı olduğu
- Nerelere seyahat ettiği
- Hangi cihazları kullandığı
- Hangi finansal işlemleri yaptığı

gibi ilişkiler de önemlidir.

Bu yaklaşım sosyal ağ analizine oldukça yakındır. Çünkü modern güvenlik ve operasyon problemleri çoğu zaman bireysel değil, ağ tabanlıdır. Örneğin organize suç yapıları veya terör ağları klasik hiyerarşik yapılardan çok, esnek ilişki ağları şeklinde çalışır. Gotham bu nedenle insanları yalnızca ayrı kayıtlar olarak değil, dinamik ilişki kümeleri olarak ele alır.

Örneğin bir analiz sırasında sistem:

- Aynı otelde kalan kişileri
- Aynı telefon ağına bağlanan cihazları
- Ortak finansal hareket yapan bireyleri
- Aynı zaman diliminde aynı bölgede bulunan kişileri

bir ilişki ağı içinde gösterebilir.

Bu yaklaşım görünmeyen bağlantıları görünür hale getirir.

Olaylar: Ağın Dinamik Yapısı

Gotham'da olaylar statik veri noktaları değildir; ağın hareket eden bileşenleridir. Çünkü gerçek dünya sürekli değişmektedir. Bir toplantı, banka transferi, sınır geçişi veya sosyal medya paylaşımı tek başına anlamlı olmayabilir. Ancak diğer olaylarla birleştiğinde operasyonel önem kazanabilir.

Örneğin tek bir uçuş kaydı önemsiz görünebilir. Ancak:

- Aynı tarihte gerçekleşen finansal transferler
- Aynı bölgede yapılan telefon görüşmeleri
- Aynı kişiyle bağlantılı diğer hareketler

ile ilişkilendirildiğinde büyük bir örüntü ortaya çıkabilir.

Bu nedenle Gotham olayları zamansal ve ilişkisel bağlam içinde değerlendirir. Burada önemli olan yalnızca “ne oldu?” sorusu değildir. Aynı zamanda:

- Kim etkiledi?
- Hangi olay hangi olayı tetikledi?
- Hangi ilişkiler tekrar ediyor?
- Ağ içinde hangi örüntüler oluşuyor?

soruları da analiz edilir.

Bu yaklaşım klasik veri analitiğinden çok daha farklıdır. Çünkü burada amaç geçmişi raporlamak değil, operasyonel anlam üretmektir.

Lokasyonlar: Coğrafyanın Semantikleşmesi

Modern dijital sistemler sürekli konum verisi üretmektedir. Telefonlar, GPS sistemleri, sensör ağları ve IoT cihazları fiziksel dünyanın dijital izlerini oluşturur. Gotham bu konum verilerini yalnızca harita üzerinde işaretlemekle kalmaz; onları ilişki ağı içinde değerlendirir.

Bir lokasyon yalnızca fiziksel bir nokta değildir. Aynı zamanda:

- İnsan hareketlerinin kesişim alanı
- Finansal faaliyet merkezi
- Sosyal etkileşim düğümü
- Operasyonel risk bölgesi

olabilir.

Örneğin sistem belirli bir bölgede:

- Olağandışı insan hareketleri
- Beklenmeyen finansal işlemler
- Belirli cihaz yoğunlukları
- Tekrarlayan ziyaret örüntüleri

tespit edebilir.

Bu yaklaşım özellikle güvenlik, lojistik ve şehir operasyonlarında önemlidir. Gotham’ın en güçlü yanlarından biri, fiziksel dünya ile dijital veri dünyasını birleştirebilmesidir. Böylece coğrafya, yalnızca mekânsal değil; semantik bir katmana dönüşür.

Finansal Hareketler: Paranın Ağ Yapısı

Modern ekonomik sistemler büyük ölçüde ağ mantığıyla çalışır. Banka transferleri, şirket ilişkileri, ticaret zincirleri ve yatırım akışları birbirine bağlı finansal ağlar oluşturur.

Gotham finansal hareketleri yalnızca muhasebe kayıtları olarak görmez. Bunun yerine şu tür ilişkileri analiz eder:

- Kim kime ödeme yaptı?
- Hangi şirketler aynı finansal örüntülere sahip?
- Hangi hesaplar aracılık düğümü olarak kullanılıyor?
- Para akışı hangi ağ üzerinden hareket ediyor?

Bu yaklaşım özellikle kara para aklama ve dolandırıcılık analizlerinde önem kazanır.

Örneğin klasik sistemlerde tek tek işlemler yasal görünebilir. Ancak bilgi grafı yaklaşımıyla analiz edildiğinde:

- Döngüsel para hareketleri
- Gizli aracı ağlar
- Şüpheli bağlantı kümeleri

ortaya çıkabilir.

Bu nedenle Gotham, finansal dünyayı doğrusal işlem listeleri yerine dinamik ilişki sistemleri olarak ele alır.

Zaman İlişkileri: Ağın Dördüncü Boyutu

Gotham'ın en önemli özelliklerinden biri zaman boyutunu ilişki ağına dahil etmesidir. Çünkü gerçek dünyadaki olayların anlamı çoğu zaman zamansal örüntüler içinde ortaya çıkar.

Örneğin:

- Bir kişinin bir lokasyonda bulunması
- Ardından belirli bir telefon görüşmesi yapması
- Sonrasında finansal işlem gerçekleştirmesi

tek başına sıradan olabilir. Ancak zaman sıralaması analiz edildiğinde operasyonel önem kazanabilir.

Bu nedenle Gotham yalnızca "kim kiminle bağlantılı?" sorusunu değil, aynı zamanda:

- Ne zaman bağlantılı oldu?
- İlişki ne kadar sürdü?

- Hangi olay hangi olaydan sonra geldi?

sorularını da inceler.

Bu yaklaşım sistemi statik bir veri haritasından çıkarıp dinamik bir operasyonel modele dönüştürür.

Gotham İçin Örnek Operasyonel Senaryo

Gotham'ın nasıl çalıştığını anlamanın en iyi yolu gerçekçi bir senaryo üzerinden ilerlemektir. Bir liman kentinde yasa dışı teknoloji kaçakçılığı şüphesi olduğunu düşünelim. Güvenlik kurumları farklı veri kaynaklarından milyonlarca kayıt toplamaktadır.

Sisteme şu tür veriler aktarılmış olsun:

Sisteme Verilen Veriler

İnsan Verileri

Kişi:

- Ahmet Kaya
- Sergey Ivanov
- Lina Chen

Telefon Verileri

Telefon kayıtları:

- Ahmet ↔ Sergey : 14 görüşme
- Sergey ↔ Lina : 22 görüşme

Seyahat Verileri

Uçuş kayıtları:

- Ahmet : İstanbul → Dubai
- Sergey : Moskova → Dubai
- Lina : Shenzhen → Dubai

Finansal Veriler

Transferler:

- Ahmet → Delta Logistics : 48.000$
- Delta Logistics → Lina Chen : 47.500$

Konum Verileri

GPS:

- Üç kişi aynı hafta Dubai Limanı çevresinde bulundu.

Kargo Verileri

Konteyner:

- Shenzhen çıkışlı elektronik ekipman
- Dubai üzerinden Doğu Avrupa'ya sevk

Gotham'ın Kurduğu Ağ

Gotham bu verileri ayrı tablolar halinde bırakmaz. Hepsini tek bir operasyonel bilgi grafında birleştirir. Ortaya çıkan ağ yaklaşık olarak şöyle görünür:

Ahmet
  ↓ CALLS
Sergey
  ↓ CALLS
Lina
  ↓ CONNECTED_TO
Delta Logistics
  ↓ SHIPS
Electronic Cargo

Ancak sistem yalnızca bağlantıları göstermekle kalmaz.

Aynı zamanda:

- zaman eşleşmelerini,
- ortak lokasyonları,
- finansal örüntüleri,
- seyahat senkronizasyonlarını

da analiz eder.

Analistin Sisteme Sorduğu Sorular

Bir Gotham analisti sisteme şu tür operasyonel sorular sorabilir:

Soru 1

"Dubai Limanı çevresinde aynı zaman diliminde bulunan kişiler kimler?"

Sistem cevabı:

Ahmet Kaya
Sergey Ivanov
Lina Chen

Soru 2

“Bu kişilerin ortak finansal bağlantıları var mı?”

Sistem cevabı:

Delta Logistics şirketi ortak düğüm olarak bulundu.

Soru 3

“Aynı ağ içinde tekrar eden iletişim örüntüsü var mı?”

Sistem cevabı:

Son 14 gün içinde:

- Ahmet ↔ Sergey : yoğun iletişim
- Sergey ↔ Lina : düzenli iletişim

Soru 4

“Bu ağ hangi fiziksel yük hareketleriyle ilişkili?”

Sistem cevabı:

Shenzhen çıkışlı elektronik konteynerler
Dubai Limanı üzerinden yönlendirilmiş.

Gotham’ın Ürettiği Operasyonel Sonuç

Tek tek bakıldığında:

- uçuş kayıtları normaldir,
- banka transferleri yasal görünebilir,
- telefon görüşmeleri sıradan olabilir.

Ancak Gotham ilişkileri üst üste bindirdiğinde yeni bir operasyonel gerçeklik ortaya çıkar. Sistem şu sonucu üretebilir:

Üç kişi, aynı zaman diliminde, aynı lokasyonda, aynı lojistik şirketiyle bağlantılıdır. Finansal transferler ve kargo hareketleri yüksek korelasyon göstermektedir. Bu, klasik veri tabanı sistemlerinin kolay kolay üretemeyeceği bir içgörüdür. Çünkü Gotham tek tek kayıtları değil, örüntüleri analiz eder.

Operasyonel Bilgi Grafının Mantığı

Gotham aslında bilgi grafı düşüncesinin operasyonel dünyaya uygulanmış biçimidir. Burada amaç yalnızca verileri depolamak değil; gerçek dünyadaki karmaşık ilişkileri görünür hale getirmektir.

Bu yaklaşım modern çağın temel problemlerine uygundur. Çünkü günümüz dünyası:

- Ağ ekonomileri
- Küresel lojistik zincirleri
- Sosyal medya ağları
- Finansal sistemler
- Siber güvenlik yapıları

gibi ilişki yoğun sistemlerden oluşur.

Dolayısıyla dünyayı anlamak için ilişkileri modellemek gerekir.

Sonuç

Palantir Gotham modern bilgi grafı çağının en önemli örneklerinden biridir. Sistem dünyayı statik veri kümeleri olarak değil; sürekli değişen ilişkiler ağı olarak görür. İnsanlar, olaylar, lokasyonlar, finansal hareketler ve zaman ilişkileri birleşerek operasyonel gerçekliğin dijital haritasını oluşturur. Bu yaklaşım yalnızca istihbarat veya güvenlik alanını değil; geleceğin şehirlerini, ekonomik sistemlerini ve yapay zekâ altyapılarını da şekillendirebilir. Gotham'ın asıl önemi belki de şuradadır: Dünyayı anlamanın yeni yolu artık nesneleri değil, ilişkileri okumaktır.

11.3 Foundry: Şirketlerin Dijital Sinir Sistemi

Modern şirketlerin en büyük problemlerinden biri veri eksikliği değildir; tam tersine veri parçalanmasıdır. Bugünün büyük kurumları inanılmaz miktarda veri üretmektedir. ERP sistemleri üretim süreçlerini izler, CRM sistemleri müşteri ilişkilerini kaydeder, IoT sensörleri makinelerden sürekli veri toplar, finans sistemleri mali hareketleri saklar ve insan kaynakları altyapıları çalışan süreçlerini yönetir. Ancak bu sistemlerin büyük bölümü birbirinden kopuktur. Veriler farklı yazılımlarda, farklı formatlarda ve farklı mantıklarla tutulur. Bu nedenle kurumlar çoğu zaman "veri zengini ama anlam fakiri" hale gelir.

Palantir Technologies tarafından geliştirilen Palantir Foundry platformu tam olarak bu problemi çözmeye çalışan sistemlerden biridir. Foundry'nin temel yaklaşımı oldukça güçlü bir fikre dayanır: Bir şirket, birbirinden bağımsız yazılımların toplamı değildir; yaşayan bir organizmadır. Eğer kurumun tüm veri akışları birbirine bağlanabilirse, şirket adeta dijital bir sinir sistemine sahip olabilir. Bu yaklaşım klasik veri ambarı (data warehouse) anlayışından farklıdır. Geleneksel sistemlerde amaç verileri merkezi bir yerde toplamak ve raporlamaktır. Foundry ise veriler arasındaki ilişkileri operasyonel bir bilgi ağına dönüştürmeye çalışır. Bu nedenle sistemin merkezinde bilgi grafı mantığı bulunur.

Kurumsal Veri Entegrasyonu

Modern şirketlerde en büyük sorunlardan biri "veri adalarıdır" (data silos). Her departman kendi yazılımını kullanır:

- Finans bölümü ERP sistemiyle çalışır

- Pazarlama CRM altyapısını kullanır
- Fabrika üretim sistemleri ayrı veri üretir
- Lojistik departmanı başka bir platform kullanır
- IoT sensörleri sürekli teknik veri gönderir

Bu parçalı yapı nedeniyle kurumun tamamını görmek zorlaşır.

Örneğin bir üretim gecikmesinin nedeni yalnızca fabrikadaki teknik problem olmayabilir. Aynı anda:

- Tedarik zinciri aksaması
- Enerji maliyetleri
- Personel eksikliği
- Lojistik darboğazı
- Talep tahmini hatası

gibi birçok unsur birlikte etkili olabilir.

Klasik sistemlerde bu veriler ayrı platformlarda bulunduğu için ilişkiler görünmez kalır. Foundry'nin yaklaşımı ise bu farklı veri kaynaklarını tek bir semantik yapı içinde ilişkilendirmektir. Bu nedenle Foundry yalnızca veri depolayan bir sistem değil, kurumsal gerçekliği modelleyen bir altyapıdır.

Burada bilgi grafı yaklaşımı kritik rol oynar. Çünkü sistem yalnızca "hangi veri var?" sorusunu değil:

- Hangi süreç hangi süreci etkiliyor?
- Hangi departman hangi operasyonla ilişkili?
- Hangi tedarik problemi hangi üretim hattını etkiliyor?
- Hangi müşteri davranışı hangi finansal sonucu doğuruyor?

sorularını da analiz eder.

ERP, CRM ve IoT Verilerinin Birleşmesi

Foundry'nin en dikkat çekici yönlerinden biri çok farklı veri türlerini aynı operasyonel model içinde birleştirebilmesidir. ERP sistemleri genellikle şirketin operasyonel omurgasını oluşturur. Üretim, satın alma, stok yönetimi ve finansal süreçler burada tutulur. Ancak ERP sistemleri çoğu zaman ilişki yoğun analizler için tasarlanmamıştır.

CRM sistemleri ise müşteri ilişkilerine odaklanır:

- Satış geçmişi

- Müşteri davranışları
- Kampanya etkileşimleri
- Servis kayıtları

IoT sistemleri ise bambaşka bir veri evreni üretir. Fabrika sensörleri, makineler, enerji sistemleri ve akıllı cihazlar sürekli gerçek zamanlı veri üretmektedir.

Klasik kurumsal mimarilerde bu üç dünya birbirinden kopuktur. Foundry ise bunları tek bir operasyonel bilgi grafi içinde birleştirmeye çalışır.

Operasyonel Senaryo: Bir Üretim Hattı Krizi

Şimdi Foundry'nin nasıl çalıştığını gerçekçi bir operasyonel senaryo üzerinden düşünelim. Bir otomotiv üretim şirketi hayal edelim. Şirketin Almanya'daki üretim tesisinde aniden üretim düşüşü yaşanıyor.

Sisteme şu veri akışları geliyor:

IoT Verileri (Fabrika Sensörleri)

- Motor 3'te titreşim artışı
- Enerji tüketiminde ani dalgalanma
- Soğutma sisteminde basınç düşüşü

ERP Verileri

- Bakım kaydı: Motor 3 için planlı bakım 2 hafta gecikmiş
- Stok: Kritik yedek parça seviyeleri düşük
- Üretim planı: Günlük hedef %18 altında

CRM Verileri

- Avrupa müşterilerinden teslimat gecikmesi şikayetleri
- Premium segment sipariş iptalleri artışı
- Servis çağrılarında yükselme

Lojistik Verileri

- Tedarik zincirinde Asya'dan gelen parça gecikmesi
- Alternatif tedarikçi kapasite yetersiz
- Depo stoklarında düzensizlik

Finans Verileri

- Üretim maliyetinde artış
- Gecikme cezaları riskinin yükselmesi
- Nakit akış baskısı

Foundry'nin Oluşturduğu Operasyonel Görünüm

Foundry tüm bu verileri ayrı tablolar olarak bırakmaz. Bunları tek bir operasyonel bilgi grafında birleştirir. Ortaya çıkan ağ şu ilişkileri üretir:

IoT Sensor → Motor 3 → Production Line → Factory
Factory → ERP Maintenance Record → Delayed Service
Supply Chain → Missing Component → Motor 3
Factory → Customer Orders → CRM Complaints
Factory → Financial Penalty Risk

Bu noktada sistem yalnızca "ne oldu?" sorusunu değil, "neden oldu?" ve "neye dönüşecek?" sorularını da yanıtlamaya başlar.

Sistem Tarafından Üretilen İçgörüler

Foundry'nin analiz çıktısı şu şekilde olabilir:

1. Kök Neden Analizi

Üretim düşüşünün ana nedeni:

- Motor 3 bileşen arızası
- Gecikmiş bakım süreci
- Tedarik zinciri gecikmesi

2. Zincir Etki Analizi

Etkilenen sistemler:
- Avrupa müşteri teslimatları
- Premium segment siparişleri
- Lojistik maliyet yapısı

3. Operasyonel Risk Haritası

Kritik risk düğümleri:
- Motor 3 (yüksek risk)
- Asya tedarik hattı (orta risk)
- Avrupa dağıtım ağı (dolaylı risk)

Operasyonel Karar Üretimi

Sistem artık yalnızca veri sunmaz; öneri üretir:

- Motor 3 için acil bakım planı oluştur
- Alternatif tedarikçiye yönel
- Avrupa teslimatlarını yeniden planla
- Kritik müşterilere proaktif bilgilendirme yap

Bu yaklaşım şirketi reaktif bir yapıdan çıkarır ve proaktif bir yapıya dönüştürür.

Dijital İkiz Perspektifi

Foundry'nin bir diğer önemli boyutu dijital ikiz yaklaşımıdır. Fabrikanın dijital ikizi şu bileşenleri içerir:

- Fiziksel makineler
- ERP süreçleri
- CRM müşteri akışları
- IoT sensörleri
- Finansal operasyonlar
- Tedarik zinciri bağlantıları

Bu yapı sayesinde sistem şu tür sorulara yanıt verebilir:

- Eğer Motor 3 tamamen durursa ne olur?
- Alternatif tedarik zinciri devreye girerse üretim ne kadar toparlanır?
- Enerji maliyetleri artarsa hangi ürünler etkilenir?

Şirketlerin Dijital Sinir Sistemi

Foundry çoğu zaman "şirketlerin dijital sinir sistemi" olarak tanımlanır. Bu benzetme oldukça anlamlıdır.

İnsan bedeninde sinir sistemi:

- Bilgi toplar
- Sinyalleri iletir
- Durumu analiz eder
- Tepki üretir

Kurumsal sistemler de aynı şekilde çalışmaya başlar.

- IoT sensörleri → sinir uçları
- ERP sistemleri → organlar

- CRM sistemleri → dış dünya etkileşimi
- Foundry → merkezi bilişsel katman

Bu yapı sayesinde şirket, kendi durumunu gerçek zamanlı olarak algılayan bir organizmaya dönüşür.

Sonuç

Palantir Foundry modern şirketlerin veri anlayışını kökten değiştiren sistemlerden biridir. Burada veri yalnızca depolanan bir kaynak değil; ilişkiler ağı içinde anlam kazanan operasyonel bir varlıktır.

ERP, CRM ve IoT sistemlerinin birleşmesiyle şirketler kendi dijital ikizlerini oluşturmaya başlamaktadır. Böylece kurumlar statik organizasyonlar olmaktan çıkıp, gerçek zamanlı öğrenen ve tepki veren dijital organizmalara dönüşmektedir. Foundry'nin asıl önemi belki de şuradadır: Geleceğin şirketleri yalnızca yazılım kullanan kurumlar değil, kendi bilişsel altyapısına sahip yaşayan bilgi ağları olacaktır.

11.4 Ontoloji Kavramının Endüstride Kullanımı

Bilgi grafı sistemlerinin yükselişiyle birlikte uzun yıllar akademik çevrelerde tartışılan bir kavram, endüstrinin merkezine yerleşmeye başladı: ontoloji. Felsefede ontoloji, "varlık nedir?" sorusunu inceleyen alan olarak bilinir.

> Bilgisayar bilimlerinde ise ontoloji, bir alan içindeki kavramların, kategorilerin ve ilişkilerin nasıl tanımlandığını ifade eder. Başka bir deyişle ontoloji, bir sistemin dünyayı nasıl anlamlandırdığını belirleyen kavramsal haritadır.

Bir zamanlar yalnızca semantik web araştırmalarının teknik konusu gibi görünen ontoloji kavramı, bugün büyük şirketlerin veri mimarisinin temel bileşenlerinden biri haline gelmiştir. Bunun nedeni oldukça açıktır: Modern kurumların problemi artık veri eksikliği değil, anlam eksikliğidir. Şirketler petabaytlarca veri toplamakta; ancak bu veriler çoğu zaman birbirinden kopuk, bağlamsız ve operasyonel olarak anlaşılması zor yapılarda bulunmaktadır.

İşte ontoloji katmanı tam bu noktada devrimsel bir rol oynar.

Ontoloji Katmanı Neden Devrim Yarattı?

Klasik veri sistemlerinde veri çoğunlukla teknik yapılara göre organize edilir. Veri tabanı tabloları, sütun isimleri, API formatları ve yazılım mimarileri ön plandadır. Ancak bu yapı teknik olmayan insanlar için oldukça karmaşıktır.

Örneğin büyük bir şirkette şu veri kaynakları olabilir:

- ERP sistemi

- CRM altyapısı
- IoT sensörleri
- İnsan kaynakları yazılımı
- Finans sistemleri
- Lojistik platformları

Her sistem kendi veri dilini kullanır. Bir yerde “müşteri” olan kavram başka sistemde “hesap sahibi” olarak geçebilir. Bir sistemde “ürün kodu” başka sistemde farklı biçimde tutulabilir. Bu durum şirket içinde büyük bir anlam karmaşası yaratır.

Ontoloji katmanı tam olarak bu problemi çözmeye çalışır. Çünkü ontoloji teknik veri yapılarını değil, gerçek dünyadaki kavramları merkeze alır.

Örneğin bir ontoloji sistemi şunu tanımlayabilir:

- “Müşteri” nedir?
- “Tedarikçi” nedir?
- “Teslimat gecikmesi” ne anlama gelir?
- “Üretim hattı” hangi süreçlerle ilişkilidir?

Böylece şirketin tüm veri sistemleri ortak bir kavramsal dil altında birleşir.

Bu yaklaşım devrimsel sonuçlar doğurmuştur. Çünkü veri artık yalnızca mühendislerin anlayabildiği teknik bir kaynak olmaktan çıkıp, kurumsal anlam sistemine dönüşmeye başlamıştır.

Verinin Anlam Katmanı

Ontoloji katmanının en önemli özelliği, veriyi “anlamsal” hale getirmesidir.

Klasik veri tabanı yaklaşımında sistemler çoğu zaman şu tür sorulara cevap verir:

- Bu tabloda kaç kayıt var?
- Hangi müşteri hangi ürünü satın aldı?
- Son üç ayın satış miktarı nedir?

Ancak ontoloji destekli sistemler çok daha karmaşık ilişkileri modelleyebilir:

- Hangi üretim problemi müşteri memnuniyetini etkiliyor?
- Tedarik zinciri gecikmesi hangi finansal riski doğuruyor?
- Belirli bir ekipman arızası hangi operasyonel süreçlerle bağlantılı?

Bu nedenle ontoloji katmanı, şirketlerin yalnızca veri depolamasını değil, kurumsal gerçekliği modellemesini sağlar.

Aslında ontoloji, şirketin “dijital dünya görüşü” haline gelir.

Teknik Olmayan İnsanların Veriyle Konuşabilmesi

Ontoloji yaklaşımının en önemli sonuçlarından biri, teknik olmayan insanların da veri sistemleriyle doğrudan etkileşime girebilmesidir.

Klasik kurumsal yapılarda veri analizi genellikle teknik uzmanların kontrolündedir:

- Veri mühendisleri
- SQL uzmanları
- BI geliştiricileri
- Veri bilimcileri

Bir yönetici veriyle ilgili soru sormak istediğinde çoğu zaman teknik ekibe ihtiyaç duyar. Örneğin: “Geçen ay yaşanan üretim gecikmelerinin müşteri şikâyetlerine etkisi nedir?” Bu soru klasik sistemlerde birçok veri kaynağının manuel olarak birleştirilmesini gerektirebilir. Ancak ontoloji katmanı sayesinde sistem gerçek dünyadaki kavramları bildiği için, kullanıcı doğal dil benzeri sorgularla veriyle iletişim kurabilir. Bu durum veri erişimini demokratikleştiriyor gibi görünmektedir.

Örneğin bir fabrika yöneticisi:

- “En riskli tedarik zinciri düğümleri hangileri?”
- “Bakım gecikmeleri hangi makinelerde yoğunlaşıyor?”
- “Enerji tüketimi ile üretim kalitesi arasında ilişki var mı?”

gibi soruları teknik bilgiye ihtiyaç duymadan sorabilir. Bu nedenle ontoloji katmanı, şirket içinde yeni bir “veri okuryazarlığı” dönemi başlatmıştır.

Yapay Zekâ ve Ontoloji

Büyük dil modellerinin yükselişiyle birlikte ontoloji sistemleri daha da önemli hale gelmiştir. Çünkü yapay zekâ sistemlerinin en büyük problemlerinden biri bağlam eksikliğidir. Dil modelleri istatistiksel örüntüler üzerinden çalışır; ancak kurumsal gerçekliği anlamakta zorlanabilir. Ontoloji katmanı burada yapay zekâya kurumsal bağlam sağlar. Örneğin bir üretim şirketinde “hat” kelimesi:

- Üretim hattı
- Telefon hattı
- Lojistik hattı

anlamına gelebilir.

Ontoloji sistemi bu kavramların bağlamsal anlamını tanımlar. Böylece yapay zekâ daha doğru operasyonel yorumlar yapabilir. Bu nedenle modern operasyonel bilgi grafı sistemlerinde ontoloji katmanı çoğu zaman "kurumsal zekânın merkezi" olarak görülmektedir.

Veri Demokrasi mi, Veri Merkezileşmesi mi?

Ontoloji tabanlı sistemler büyük bir paradoks da yaratmaktadır. Bir yandan bu sistemler veri erişimini demokratikleştiriyor gibi görünür. Çünkü teknik olmayan insanlar da veriyle etkileşime girebilir hale gelir. Ancak diğer yandan tüm kurumsal gerçekliğin tek bir ontolojik model altında birleşmesi, büyük ölçüde merkezileşmiş bir kontrol yapısı oluşturabilir.

Bu durum önemli sorular doğurur:

- Ontolojiyi kim tasarlıyor?
- Hangi kavramlar sisteme dahil ediliyor?
- Hangi ilişkiler görünür hale geliyor?
- Hangi anlam çerçevesi "resmî gerçeklik" haline geliyor?

Çünkü ontoloji yalnızca teknik bir model değildir; aynı zamanda bir dünya görüşüdür.

Örneğin bir şirket çalışanlarını yalnızca "verimlilik düğümleri" olarak modelleyebilir. Başka bir kurum ise insan ilişkilerini ve sosyal etkileşimleri daha fazla önemseyebilir.

Dolayısıyla ontoloji tasarımı aslında politik bir süreçtir.

Operasyonel Güç ve Semantik Kontrol

Ontoloji katmanı güç üretir. Çünkü bir sistemi tanımlayan kavramları kontrol etmek, o sistemi yorumlama biçimini de kontrol etmek anlamına gelir. Bu nedenle büyük teknoloji şirketleri ve operasyonel platformlar ontoloji sistemlerine büyük yatırım yapmaktadır.

Palantir Technologies gibi şirketlerin geliştirdiği operasyonel bilgi grafı sistemlerinde ontoloji, veri entegrasyonunun ötesinde bir işlev görür. Sistem yalnızca verileri bağlamaz; kurumsal gerçekliği yeniden organize eder. Aslında bu yaklaşım geleceğin dijital ekonomisinde çok kritik olabilir. Çünkü yapay zekâ çağında rekabet yalnızca veri miktarıyla değil, anlam üretme kapasitesiyle de belirlenecektir.

Ontoloji kavramı, akademik semantik web çalışmalarından çıkarak modern endüstrinin merkezine yerleşmiştir. Bunun nedeni yalnızca veri entegrasyonu değildir. Ontoloji katmanı, kurumların kendi operasyonel gerçekliklerini anlamlandırmasını sağlayan bilişsel altyapıya dönüşmektedir.

Teknik olmayan insanların veriyle konuşabilmesi, kurumsal süreçlerin semantik hale gelmesi ve yapay zekâ sistemlerinin bağlamsal zekâ kazanması bu dönüşümün temel sonuçlarıdır.

Ancak ontoloji aynı zamanda güç üretir. Çünkü dünyayı nasıl tanımladığınız, onu nasıl yöneteceğinizi de belirler. Bu nedenle ontoloji çağında asıl mücadele yalnızca veri üzerinde değil, anlamın kontrolü üzerinde gerçekleşecektir.

11.5 Bilgi Grafı + Yapay Zekâ + Operasyon

Son yıllarda veri teknolojilerinin geçirdiği en önemli dönüşümlerden biri, bilgi grafı sistemlerinin yalnızca "analiz araçları" olmaktan çıkıp doğrudan "operasyonel karar motorları" haline gelmesidir. Bu dönüşüm, yapay zekâ sistemlerinin gelişimiyle birleştiğinde çok daha güçlü bir yapıya evrilmiştir: bilgi grafı + yapay zekâ + operasyon üçlüsü.

Bu yeni yaklaşımda amaç artık yalnızca geçmişi anlamak değildir. Amaç, gerçek zamanlı olarak dünyayı algılamak, yorumlamak ve aksiyon üretebilmektir. Yani sistemler artık sadece "raporlayan" değil, aynı zamanda "karar veren" yapılara dönüşmektedir.

Palantir Technologies gibi şirketlerin geliştirdiği operasyonel platformlar bu dönüşümün en somut örneklerindendir. Bu tür sistemlerde bilgi grafı, yapay zekânın hafızası; yapay zekâ ise bilgi grafının akıl yürütme katmanı gibi çalışır. Operasyon ise bu iki katmanın gerçek dünyaya temas ettiği noktadır.

Sadece analiz değil, karar verme

Geleneksel veri analitiği sistemleri genellikle geçmişe bakar. "Ne oldu?", "Neden oldu?" ve "Ne kadar oldu?" gibi sorulara yanıt üretir. Ancak modern dünyada bu yeterli değildir. Çünkü birçok sistemde kararlar milisaniyeler içinde verilmek zorundadır.

Örneğin:

- Finansal piyasalarda risk yönetimi
- Lojistik ağlarda rota optimizasyonu
- Enerji şebekelerinde yük dengeleme
- Siber güvenlikte saldırı tespiti
- Askeri ve istihbari operasyonlarda tehdit analizi

gibi alanlarda sadece analiz yapmak yeterli değildir; karar üretmek gerekir.

Bilgi grafı bu noktada kritik bir rol oynar. Çünkü karar verme süreci yalnızca tek bir veri noktasına değil, ilişkiler ağına dayanır.

Örneğin bir lojistik sisteminde gecikme kararı verilecekse:

- Tedarik zinciri durumu
- Alternatif rota seçenekleri
- Maliyet etkileri

- Müşteri öncelikleri
- Operasyonel kapasite

aynı anda değerlendirilmelidir.

Yapay zekâ bu noktada devreye girerek bilgi grafı üzerinde akıl yürütme yapabilir. Böylece sistem yalnızca "durumu açıklayan" değil, "aksiyon öneren" bir yapıya dönüşür.

Gerçek zamanlı durum farkındalığı

Modern operasyonların en kritik ihtiyacı "gerçek zamanlı durum farkındalığıdır" (real-time situational awareness). Çünkü dünya artık statik değildir; sürekli hareket eden bir ağdır.

Sensörler, mobil cihazlar, finansal sistemler, sosyal medya akışları ve kurumsal yazılımlar sürekli veri üretir. Ancak bu veri tek başına anlamlı değildir. Önemli olan bu verilerin ilişkisel olarak yorumlanmasıdır.

Bilgi grafı sistemleri burada güçlü bir avantaj sağlar. Çünkü:

- Varlıkları (insanlar, makineler, kurumlar) düğüm olarak temsil eder
- İlişkileri bağlantılar olarak modeller
- Zaman boyutunu dinamik bir katman olarak ekler

Bu yapı sayesinde sistem yalnızca "şu anda ne oluyor?" sorusuna değil, aynı zamanda:

- "Ne değişiyor?"
- "Ne hızla değişiyor?"
- "Hangi değişim hangi değişimi tetikliyor?"

sorularına da yanıt verebilir.

Örneğin bir üretim tesisinde:

- Bir makinede sıcaklık artışı
- Aynı anda enerji tüketiminde yükselme
- Üretim kalitesinde düşüş
- Bakım geçmişinde gecikme

gibi olaylar birlikte analiz edildiğinde, sistem potansiyel bir arıza riskini önceden tahmin edebilir.

Bu, yalnızca veri izleme değil; operasyonel farkındalık üretimidir.

İnsan-makine ortak aklı

Bilgi grafi ve yapay zekâ entegrasyonunun en ileri aşaması, insan-makine ortak aklı (human-machine collective intelligence) kavramıdır. Bu modelde insan ve yapay zekâ birbirinin yerine geçmez; birbirini tamamlar.

İnsanlar:

- Stratejik düşünme
- Etik değerlendirme
- Bağlamsal yorumlama
- Belirsizlik yönetimi

gibi alanlarda güçlüdür.

Yapay zekâ ise:

- Büyük veri analizi
- Hızlı ilişki keşfi
- Örüntü tanıma
- Gerçek zamanlı hesaplama

gibi alanlarda üstünlük sağlar.

Bilgi grafi bu iki dünyayı birleştiren köprü görevi görür. Çünkü insanın kavramsal düşüncesi ile makinenin sayısal işleme kapasitesi aynı semantik yapı içinde buluşur.

Örneğin bir kriz yönetim senaryosunda:

- Sistem binlerce veri noktasını analiz eder
- Olası senaryoları üretir
- Risk haritaları oluşturur
- Müdahale önerileri sunar

İnsan operatör ise bu önerileri:

- Stratejik hedefler
- Etik sınırlar
- Politik öncelikler

çerçevesinde değerlendirir.

Bu yapı “tam otomasyon” değil, “artırılmış zekâ” modelidir.

Operasyonel bilgi grafının yeni rolü

Geleneksel bilgi grafı sistemleri çoğunlukla bilgi arama ve analiz için kullanılırdı. Ancak yeni nesil sistemlerde bilgi grafı artık bir "operasyon çekirdeği" haline gelmektedir.

Bu çekirdek:

- Veriyi toplar
- İlişkileri kurar
- Bağlamı oluşturur
- Yapay zekâya düşünme zemini sağlar
- Operasyonel sistemlere karar önerir

Bu nedenle bilgi grafı artık pasif bir veri yapısı değil, aktif bir bilişsel altyapıdır. Palantir Technologies gibi platformlar bu dönüşümün erken örneklerini göstermiştir. Ancak gelecekte bu yaklaşım yalnızca büyük kurumlarla sınırlı kalmayabilir; şehir yönetimlerinden sağlık sistemlerine, finansal ağlardan bilimsel araştırmalara kadar geniş bir alana yayılabilir.

Sonuç

Bilgi grafı + yapay zekâ + operasyon birleşimi, veri teknolojilerinin en ileri aşamasını temsil etmektedir. Bu yapı yalnızca analiz yapan sistemlerden çok daha fazlasıdır; gerçek zamanlı karar verebilen, durum farkındalığı geliştiren ve insanla birlikte çalışan hibrit zekâ sistemleridir.

Bu yeni paradigmada dünya artık sadece gözlemlenen bir yer değil, aynı zamanda hesaplanan, simüle edilen ve müdahale edilen bir ağ haline gelmektedir. İnsan-makine ortak aklı ise bu ağın en kritik karar mekanizmasıdır.

11.6 Savunma Sanayi ve Bilgi Ağları

Savunma sanayi, bilgi teknolojilerinin en yoğun ve en kritik biçimde kullanıldığı alanlardan biridir. Özellikle son yirmi yılda savaşın doğası köklü bir değişim geçirmiştir. Artık savaş alanı yalnızca fiziksel bir coğrafya değildir; aynı zamanda devasa bir veri ve ilişki ağıdır. Bu yeni ortamda başarı, ateş gücünden çok bilgi üstünlüğüne bağlı hale gelmiştir. Bu nedenle modern savunma sistemleri giderek daha fazla "bilgi grafı" ve yapay zekâ tabanlı mimarilere yönelmektedir.

Bu dönüşümün temelinde şu fikir vardır: Modern çatışmalar, doğrusal olaylar değil, karmaşık ağlardır. İnsanlar, araçlar, sensörler, iletişim sistemleri ve finansal akışlar birbirine bağlı düğümlerden oluşur. Bu düğümler arasındaki ilişkileri anlamadan sahadaki durumu anlamak mümkün değildir.

Palantir Technologies gibi şirketler bu yaklaşımı operasyonel bilgi grafı sistemleriyle savunma alanına taşımıştır. Bu sistemlerde amaç yalnızca veri toplamak değil, veriyi anlamlı ilişki ağlarına dönüştürerek gerçek zamanlı operasyonel farkındalık üretmektir.

İstihbarat sistemleri: parçalı veriden ağ zekâsına

Geleneksel istihbarat sistemleri uzun süre boyunca parçalı veri toplama modeline dayanıyordu. Farklı kurumlar farklı türde veriler üretir:

- Sinyal istihbaratı (SIGINT)
- İnsan istihbaratı (HUMINT)
- Görüntü istihbaratı (IMINT)
- Siber istihbarat
- Açık kaynak istihbaratı (OSINT)

Ancak bu veriler çoğu zaman ayrı sistemlerde tutulur ve birbirleriyle doğrudan ilişkilendirilmez.

Modern yaklaşım ise bu verileri tek bir bilgi grafi içinde birleştirmektir. Böylece her veri parçası yalnızca kendi başına değil, diğer verilerle olan ilişkisi üzerinden anlam kazanır.

Örneğin:

- Bir telefon görüşmesi
- Bir finansal transfer
- Bir sınır geçişi
- Bir sosyal medya etkileşimi

tek başına sıradan görünebilir. Ancak bu olaylar aynı zaman diliminde ve aynı kişi ağlarıyla ilişkiliyse, daha büyük bir operasyonel örüntü ortaya çıkabilir.

İşte bilgi grafı burada devreye girer: ilişkileri görünür hale getirir.

Savaş alanı verisi: gerçek zamanlı karmaşıklık

Modern savaş alanı artık statik değildir. Gerçek zamanlı olarak değişen çok katmanlı bir veri ortamı vardır. Bu ortamda:

- Kara birlikleri
- Hava araçları
- Deniz platformları
- Uydu sistemleri
- Siber operasyonlar
- Lojistik destek ağları

aynı anda veri üretir.

Bu verilerin tek bir merkezde toplanması yeterli değildir. Çünkü sorun veri miktarı değil, veri ilişkilerinin karmaşıklığıdır.

Bilgi grafı sistemleri savaş alanını bir "ilişki haritası" olarak modelleyebilir. Örneğin:

- Bir birliğin konumu
- Yakınındaki düşman hareketleri
- İletişim sinyalleri
- Hava durumu koşulları
- Lojistik ikmal hatları

birlikte analiz edilerek operasyonel kararlar üretilebilir.

Bu yaklaşım savaş alanını bir "canlı sistem" haline getirir. Sistem sürekli değişir ve sürekli yeniden hesaplanır.

Drone ve sensör ağları: dijital algı katmanı

Modern savunma sistemlerinin en önemli bileşenlerinden biri de drone ve sensör ağlarıdır. Bu sistemler, savaş alanının dijital algı katmanını oluşturur.

Drone'lar görüntü ve video verisi üretirken, sensörler:

- Hareket
- Isı
- Ses
- Radar sinyalleri
- Elektronik izler

gibi çok farklı veri türlerini toplar.

Ancak bu veriler tek başına anlamlı değildir. Asıl değer, bu verilerin birbirleriyle ilişkilendirilmesinden doğar.

Bilgi grafı burada kritik bir rol oynar. Çünkü:

- Her drone bir düğüm
- Her sensör bir veri kaynağı
- Her tespit bir olay
- Her hareket bir ilişki

olarak modellenebilir.

Örneğin bir bölgede:

- Termal anomali
- Hareketli araç tespiti
- Radyo sinyali artışı

aynı anda gözlemleniyorsa, sistem potansiyel bir operasyonel aktiviteyi tespit edebilir.

Bu yapı sayesinde savaş alanı yalnızca izlenen bir yer değil, sürekli analiz edilen bir ağ haline gelir.

Hedef ilişkilendirme sistemleri

Modern savunma teknolojilerinin en kritik problemlerinden biri hedef ilişkilendirmedir. Çünkü bir "hedef" genellikle tek bir veri noktasından ibaret değildir. Bir hedef:

- Bir kişi olabilir
- Bir araç olabilir
- Bir iletişim ağı olabilir
- Bir finansal yapı olabilir
- Ya da bunların hepsinin birleşimi olabilir

Bu nedenle hedef tanımı statik değil, ilişkisel bir yapıdır.

Bilgi grafı sistemleri bu problemi çözmek için geliştirilmiştir. Çünkü hedefler:

- Davranış örüntülerine
- Sosyal bağlantılara
- Coğrafi hareketlere
- Finansal ilişkilere
- İletişim ağlarına

göre modellenebilir.

Örneğin bir hedef sistemde şu şekilde temsil edilebilir:

- Birden fazla kimlik
- Birden fazla lokasyon
- Birden fazla cihaz
- Birden fazla finansal hesap

Bu parçaların birleşimi gerçek hedef profilini oluşturur.

Bu nedenle modern savunma sistemlerinde “kimlik” tekil bir kavram olmaktan çıkmış, ağ tabanlı bir yapı haline gelmiştir.

Operasyonel bilgi grafı ve savunma zekâsı

Savunma sanayinde bilgi grafı sistemlerinin en önemli katkısı, operasyonel zekâ üretmesidir. Bu sistemler yalnızca veriyi depolamaz; aynı zamanda:

- İlişki keşfi yapar
- Anomali tespit eder
- Risk analizi üretir
- Senaryo simülasyonu yapar
- Karar destek mekanizmaları oluşturur

Bu nedenle modern savunma sistemleri giderek daha fazla “öngörücü” hale gelmektedir.

Örneğin bir sistem:

- Bir saldırı olasılığını önceden tahmin edebilir
- Lojistik zayıflıkları belirleyebilir
- İletişim ağlarındaki kırılmaları analiz edebilir

Bu, klasik istihbarat modelinden çok daha ileri bir aşamadır.

Etik ve politik boyut

Savunma alanında bilgi grafı sistemlerinin yükselişi önemli etik soruları da beraberinde getirir. Çünkü bu sistemler:

- Büyük ölçekli veri toplama
- Kapsamlı ilişki analizi
- Davranış tahmini
- Gerçek zamanlı izleme

kapasitelerine sahiptir.

Bu durum mahremiyet, gözetim ve güç dengesi açısından ciddi tartışmalar yaratır.

Bu nedenle savunma sanayinde bilgi grafı kullanımı yalnızca teknik bir konu değil, aynı zamanda politik ve etik bir meseledir.

Sonuç

Savunma sanayi ve bilgi ağları arasındaki ilişki, modern savaş ve güvenlik anlayışını kökten değiştirmektedir. İstihbarat sistemlerinden savaş alanı verisine, drone ağlarından hedef ilişkilendirme sistemlerine kadar her şey artık birer bilgi grafı problemi haline gelmiştir.

Palantir Technologies gibi şirketlerin geliştirdiği sistemler, bu dönüşümün erken örneklerini temsil etmektedir. Gelecekte savunma sistemleri yalnızca güçlü değil, aynı zamanda "akıllı", "bağlantısal" ve "öngörücü" olacaktır.

Bu yeni çağda üstünlük, en çok veriye sahip olanın değil; veriyi en iyi ilişkilendirenin olacaktır.

11.7 Sağlık, Enerji ve Üretimde Palantir Yaklaşımı

Modern endüstriyel sistemlerin ortak problemi giderek daha görünür hale gelmiştir: veri çokluğu değil, veri parçalanması. Sağlık, enerji ve üretim gibi kritik sektörlerde milyonlarca veri noktası üretilmesine rağmen, bu verilerin büyük kısmı birbirinden kopuk sistemlerde yaşamaktadır. Bu kopukluk, karar alma süreçlerini yavaşlatmakta ve kriz anlarında ciddi kör noktalar oluşturmaktadır. Bu sorunu çözmeye yönelik en kapsamlı yaklaşımlardan biri, Palantir Technologies tarafından geliştirilen operasyonel bilgi grafı modelidir. Bu yaklaşım, veriyi yalnızca analiz edilen bir kaynak olarak değil, sürekli çalışan bir karar sistemi olarak ele alır.

Bu modelin temel fikri şudur: Sağlık sistemleri, enerji ağları ve üretim zincirleri aslında birbirine bağlı devasa bilgi graflarıdır. Eğer bu grafler gerçek zamanlı olarak modellenebilir ve güncellenebilirse, sistemler yalnızca geçmişi raporlayan yapılar olmaktan çıkar; aynı zamanda geleceği öngören ve müdahale eden yapılar haline gelir.

COVID veri yönetimi: kriz döneminde bilgi grafı

COVID-19 pandemisi, modern veri sistemlerinin hem gücünü hem de zayıflığını açık biçimde ortaya koymuştur. Pandemi sürecinde devletler ve sağlık kurumları büyük miktarda veri üretmiştir:

- Hasta kayıtları
- Test sonuçları
- Aşı dağıtım verileri
- Hastane doluluk oranları
- Coğrafi yayılım haritaları
- Temas takibi verileri

Ancak bu verilerin büyük kısmı farklı sistemlerde, farklı formatlarda ve farklı hızlarda tutulmuştur. Bu durum, kriz yönetimini zorlaştırmıştır.

Bilgi grafı yaklaşımı bu noktada kritik bir avantaj sunar. Çünkü pandemi verileri yalnızca sayısal listeler olarak değil, ilişkisel bir ağ olarak modellenebilir:

- Bir hasta → diğer hastalarla temas ilişkisi
- Bir şehir → vaka yoğunluğu ilişkisi
- Bir hastane → kapasite ve kaynak ilişkisi
- Bir mutasyon → yayılım hız ilişkisi

Bu ilişkiler bir araya geldiğinde, sistem yalnızca mevcut durumu göstermekle kalmaz; aynı zamanda yayılım senaryolarını da simüle edebilir.

Örneğin bir bölgede vaka artışı başladığında sistem şu tür ilişkileri analiz edebilir:

- Yakın şehirlerdeki hareketlilik
- Ulaşım ağlarındaki yoğunluk
- Hastane kapasite sınırları
- Aşılanma oranları

Bu sayede sağlık yönetimi reaktif değil, proaktif hale gelir.

COVID süreci, sağlık verisinin yalnızca tıbbi değil, aynı zamanda operasyonel bir ağ problemi olduğunu göstermiştir.

Fabrika optimizasyonu: üretim ağının dijital modeli

Üretim sistemleri, modern ekonominin en karmaşık ağlarından biridir. Bir fabrikanın içinde bile yüzlerce makine, binlerce süreç ve sürekli değişen üretim akışları vardır. Bu yapı çoğu zaman klasik yazılımlarla tam olarak modellenemez.

Palantir yaklaşımı, fabrikayı bir bilgi grafı olarak ele alır. Bu modelde:

- Makineler düğümler
- Üretim hatları ilişkiler
- Sensörler veri kaynakları
- Operatörler etkileşim noktaları

olarak temsil edilir.

Bu sayede üretim süreci yalnızca “ne üretildi?” sorusu üzerinden değil, “nasıl üretildi?” sorusu üzerinden analiz edilir.

Örneğin bir üretim hattında kalite düşüşü yaşandığında sistem şunları aynı anda analiz edebilir:

- Hangi makine daha önce bakım görmüş
- Hangi vardiyada üretim yapılmış
- Hangi hammadde partisi kullanılmış

- Enerji dalgalanmaları olmuş mu
- Operatör değişimi var mı

Bu ilişkisel analiz sayesinde sorun yalnızca tespit edilmez, aynı zamanda kök nedenleri de ortaya çıkar.

Ayrıca bilgi grafı modeli üretim süreçlerini optimize etmek için de kullanılır. Sistem:

- Darboğazları belirleyebilir
- Enerji kullanımını optimize edebilir
- Makine kullanım oranlarını dengeleyebilir
- Üretim planlamasını dinamik olarak değiştirebilir

Bu nedenle fabrika artık statik bir üretim alanı değil, sürekli öğrenen bir dijital organizmaya dönüşür.

Tedarik zinciri görünürlüğü: küresel ağın haritalanması

Modern tedarik zincirleri küresel ölçekte karmaşık ağlardır. Bir ürünün üretimi için:

- Hammaddeler farklı ülkelerden gelir
- Üretim farklı tesislerde yapılır
- Lojistik farklı taşıyıcılar üzerinden gerçekleşir
- Dağıtım farklı pazarlara yayılır

Bu yapı, küçük bir aksaklığın bile küresel etkiler yaratmasına neden olur.

Bilgi grafı yaklaşımı tedarik zincirini uçtan uca görünür hale getirir. Bu modelde:

- Tedarikçiler düğümler
- Nakliye rotaları ilişkiler
- Depolar ara noktalar
- Ürünler hareket eden varlıklar

olarak temsil edilir.

Bu sayede sistem şu sorulara gerçek zamanlı yanıt verebilir:

- Hangi tedarikçi darboğaz yaratıyor?
- Hangi rota gecikmeye neden oluyor?
- Hangi ürün stok riski altında?
- Hangi alternatif tedarik zinciri kullanılabilir?

Örneğin bir limanda yaşanan gecikme, yalnızca o bölgeyi değil, tüm üretim ağını etkileyebilir. Bilgi grafı sistemi bu tür zincirleme etkileri önceden simüle edebilir.

Bu yaklaşım özellikle küresel kriz dönemlerinde büyük önem kazanır. Pandemi, savaşlar ve doğal afetler tedarik zincirlerinin ne kadar kırılgan olduğunu göstermiştir.

Operasyonel zekâya geçiş

Sağlık, enerji ve üretim alanlarındaki Palantir yaklaşımının ortak noktası, "operasyonel zekâ" üretmesidir. Bu kavram, yalnızca veri analizi değil, doğrudan aksiyon üretme kapasitesini ifade eder.

Geleneksel sistemler:

- Veri toplar
- Rapor üretir
- İnsan karar verir

Operasyonel bilgi grafı sistemleri ise:

- Veriyi toplar
- İlişkileri analiz eder
- Senaryo üretir
- Karar önerir
- Hatta bazı durumlarda otomatik aksiyon alır

Bu nedenle insan ile makine arasındaki ilişki değişir. İnsan artık veri yorumlayan kişi değil, sistemin ürettiği önerileri yöneten stratejik karar verici haline gelir.

Risk, etik ve bağımlılık

Bu yaklaşım büyük avantajlar sağlasa da bazı riskler de içerir. Özellikle:

- Veri bağımlılığı
- Merkezi karar yapıları
- Algoritmik önyargılar
- Mahremiyet sorunları

önemli tartışma alanlarıdır.

Sağlık verileri, üretim süreçleri ve tedarik zinciri bilgileri son derece hassas verilerdir. Bu nedenle bu sistemlerin nasıl tasarlandığı ve kimler tarafından kontrol edildiği kritik bir sorudur.

Sonuç

Sağlık, enerji ve üretim sektörlerinde bilgi grafı yaklaşımı yalnızca bir teknoloji değil, yeni bir operasyonel paradigma haline gelmiştir. Palantir Technologies tarafından geliştirilen sistemler, veri parçalanması problemini çözerek kurumları gerçek zamanlı çalışan dijital organizmalara dönüştürmektedir.

COVID veri yönetiminden fabrika optimizasyonuna, tedarik zinciri görünürlüğünden kriz yönetimine kadar tüm alanlarda ortak bir dönüşüm yaşanmaktadır: sistemler artık veriyi yalnızca saklamamakta, onu ilişkilendirerek karar üretmektedir.

Bu dönüşümün özü şudur: Geleceğin rekabet avantajı veri miktarında değil, veri ilişkilerini anlayabilme kapasitesinde yatmaktadır.

11.8 Palantir ve Yapay Zekâ Platformları

Yapay zekâ teknolojilerinin son yıllarda geçirdiği dönüşüm, yalnızca model kapasitesindeki artışla açıklanamaz. Asıl kırılma noktası, yapay zekânın “laboratuvar modeli” olmaktan çıkıp “operasyonel sistem” haline gelmesidir. Büyük dil modelleri (LLM’ler) metin üretme, özetleme ve sınıflandırma gibi görevlerde olağanüstü başarı gösterirken, kurumların gerçek ihtiyacı bu modelleri doğrudan iş süreçlerine bağlayabilmektir. Bu noktada Palantir Technologies tarafından geliştirilen Yapay Zekâ Platformu (AIP), yapay zekâyı bilgi grafı tabanlı operasyonel sistemlerle birleştiren yeni bir yaklaşım sunar.

AIP’nin temel amacı, yapay zekâyı yalnızca “cevap veren bir araç” olmaktan çıkarıp “karar veren ve eylem üreten bir sistem” haline getirmektir. Bu yaklaşımda yapay zekâ, kurumun veri ekosistemiyle doğrudan entegre edilir ve bilgi grafı üzerinden tüm operasyonel bağlamı görür. Böylece model yalnızca dilsel örüntülere değil, gerçek dünyadaki ilişkilere dayanarak karar üretir. Palantir'in Gotham ve AIP gibi platformlarının İsrail'in Gazze, Lübnan ve İran'a yönelik operasyonlarında hedef belirleme ve veri analizi için kullanıldığını gösteriyor.

AIP (Yapay Zekâ Platformu): Operasyonel zekânın çekirdeği

AIP, klasik yapay zekâ uygulamalarından farklı olarak bir “model platformu” değil, bir “operasyon platformu”dur. Buradaki temel fark şudur: Geleneksel yapay zekâ sistemleri genellikle bağımsız modeller olarak çalışır. Bir veri seti üzerinde eğitilir ve belirli bir görevi yerine getirir. Ancak AIP yaklaşımında yapay zekâ, sürekli güncellenen bir bilgi grafı ile birlikte çalışır.

Bu yapı sayesinde sistem:

- Gerçek zamanlı veri akışını izler
- Kurumsal ilişkileri anlar
- Operasyonel bağlamı korur
- Aksiyon üretebilir

Örneğin bir lojistik şirketinde AIP, yalnızca teslimat gecikmesini tespit etmekle kalmaz; aynı zamanda alternatif rota önerileri oluşturabilir, stok dağılımını yeniden planlayabilir ve riskli bölgeleri işaretleyebilir.

Bu nedenle AIP, yapay zekâyı bir “karar destek sistemi” olmaktan çıkarıp “karar üretim sistemi” haline getirir.

LLM entegrasyonu: dil modellerinin operasyonelleşmesi

Büyük dil modelleri (LLM’ler), doğal dil işleme alanında devrim yaratmıştır. Ancak bu modeller tek başına kullanıldığında genellikle bağlam eksikliği yaşar. Çünkü LLM’ler dünyayı doğrudan “ilişkisel bir yapı” olarak değil, istatistiksel örüntüler üzerinden temsil eder.

AIP yaklaşımında LLM’ler, bilgi grafı ile entegre edilerek bu eksiklik giderilir. Bu entegrasyon sayesinde:

- LLM’ler yalnızca metin üretmez
- Aynı zamanda kurumsal veriye bağlanır
- Gerçek zamanlı bilgi grafı üzerinden sorgu yapar
- Operasyonel kararları yorumlar

Örneğin bir finans kurumunda kullanıcı şu soruyu sorabilir:

“Son üç ayda yüksek riskli işlem artışı nerede yoğunlaştı?”

LLM tek başına bu soruya güvenilir cevap veremez. Ancak bilgi grafı ile entegre edildiğinde:

- Hesap hareketleri
- Kullanıcı bağlantıları
- Coğrafi dağılımlar
- Zaman serileri

üzerinden analiz yaparak anlamlı sonuç üretebilir. Bu yapı, LLM’leri “metin üretici” olmaktan çıkarıp “kurumsal zekâ arayüzü” haline getirir.

Palantir'in teknolojisi, Ukrayna savaşında sadece bir araç olarak değil, aynı zamanda bir "veri laboratuvarı" olarak da kullanılıyor. Bu, bir yapay zekâ modelinin gerçek bir savaşın acımasız verileriyle nasıl eğitildiğini gösteren ilginç bir vakadır.

- **Dataroom Projesi:** Ukrayna hükümeti, Palantir ile birlikte **Dataroom** adı verilen güvenli bir dijital ortam oluşturdu. Bu ortamın amacı, Ukraynalı savunma şirketlerinin yapay zekâ modellerini eğitmek için **gerçek savaş verilerini** (Rus hava saldırıları, Şahit tipi insansız hava araçlarının uçuş verileri gibi) kullanmasına olanak sağlamak.

- **Hedef:** Bu sayede özellikle önleme drone'larının hedef tespiti, sınıflandırması ve etkisiz hale getirme kabiliyetleri geliştiriliyor. Bu proje kapsamında şu anda 100'den fazla şirket, 80 farklı yapay zekâ modelini Ukrayna'nın sağladığı savaş verileriyle eğitiyor.

Birleşmiş Milletler Özel Raportörü Francesca Albanese tarafından hazırlanan rapor, **Palantir'i sürecin bir parçası olmakla** doğrudan suçluyor. Suçlamalar şu yönde:

- **Teknoloji Altyapısı:** Palantir'in Gotham ve AIP (Yapay Zeka Platformu) gibi platformları, İsrail ordusunun bu devasa veriyi işlemesine ve yapay zeka kararlarını sahaya entegre etmesine olanak sağlıyor.
- **"Where's Daddy?" ve "Lavender"a Destek:** Raporda, Palantir'in **"Where's Daddy?"** ve **"Lavender"** gibi sistemlerin çalışması için gerekli olan tahmine dayalı polislik (predictive policing) ve otomatik karar destek mekanizmalarını sağladığı belirtiliyor.
- **Stratejik Ortaklık:** Palantir'in CEO'su Alex Karp, Ocak 2024'te Tel Aviv'de bir toplantı yaparak İsrail ordusu ile stratejik bir anlaşma imzalamıştır.

Palantir, bu suçlamaları reddetmektedir. Şirket, İsrail ile olan faaliyetlerinin 7 Ekim 2023 öncesine dayandığını ve ABD'nin diğer müttefiklerine verilen standart desteğin ötesine geçmediğini savunmaktadır.

Operasyonel ajan sistemleri: yapay zekânın eylem katmanı

AIP mimarisinin en ileri aşaması operasyonel ajan sistemleridir. Bu sistemler, yapay zekâ modellerinin yalnızca öneri sunmasını değil, belirli sınırlar içinde aksiyon almasını sağlar.

Bir operasyonel ajan şu bileşenlerden oluşur:

- Bilgi grafı (kurumsal bağlam)
- LLM (yorumlama ve akıl yürütme)
- Kural motorları (etik ve güvenlik sınırları)
- API bağlantıları (gerçek sistemlere erişim)

Bu yapı sayesinde ajanlar yalnızca “ne yapılmalı?” sorusuna cevap vermez; aynı zamanda “nasıl yapılmalı?” sorusunu da çözer.

Örneğin bir enerji şirketinde operasyonel ajan:

- Şebeke yükünü izler
- Talep artışını tahmin eder
- Riskli bölgeleri belirler
- Gerekirse otomatik dengeleme işlemi başlatır

Bu süreçte insan müdahalesi tamamen ortadan kalkmaz; ancak insan artık mikro kararlar yerine makro stratejilere odaklanır.

Bilgi grafı + yapay zekâ birleşimi

AIP'nin en önemli yeniliği, bilgi grafı ile yapay zekâyı aynı operasyonel katmanda birleştirmesidir. Bu birleşim sayesinde sistem yalnızca veri analizi yapmaz; aynı zamanda anlam üretir.

Bilgi grafı:

- Varlıkları temsil eder
- İlişkileri modeller
- Bağlam oluşturur

Yapay zekâ ise:

- Bu yapıyı yorumlar
- Sorgular
- Tahmin üretir
- Karar verir

Bu iki yapı birleştiğinde ortaya çıkan sistem, klasik veri analitiğinin çok ötesine geçer.

Örneğin bir üretim tesisinde:

- Makine arızası
- Tedarik gecikmesi
- Enerji maliyeti artışı
- Talep dalgalanması

aynı anda analiz edilerek bütünsel bir operasyon planı oluşturulabilir.

İnsan-makine işbirliği modeli

AIP yaklaşımının en kritik yönlerinden biri, insan-makine işbirliğini yeniden tanımlamasıdır. Burada amaç insanı devre dışı bırakmak değildir. Tam tersine insanın bilişsel kapasitesini artırmaktır.

İnsan:

- Stratejik karar verir
- Etik sınırları belirler
- Belirsizlikleri yorumlar

Yapay zekâ:

- Veri analiz eder
- Alternatif senaryolar üretir
- Riskleri hesaplar
- Öneriler sunar

Bu nedenle ortaya çıkan yapı “hibrit zekâ”dır.

Bu modelde insan, sistemin dışında değil; sistemin içinde yer alır. Yapay zekâ ise insanın karar verme sürecini genişleten bir bilişsel uzantı haline gelir.

Operasyonel dönüşüm: analizden eyleme

Geleneksel veri sistemleri genellikle “analiz odaklıdır”. Ancak AIP yaklaşımı “eylem odaklıdır”. Bu fark çok kritiktir.

Analitik sistemler:

- Veri toplar
- Rapor üretir
- İnsan yorum yapar

Operasyonel yapay zekâ sistemleri ise:

- Veriyi sürekli analiz eder
- Bağlamı anlar
- Karar önerir
- Gerekirse eyleme geçer

Bu dönüşüm, kurumların çalışma şeklini kökten değiştirir.

Riskler ve kontrol mekanizmaları

Bu kadar güçlü sistemler doğal olarak bazı riskler de içerir:

- Aşırı otomasyon
- Yanlış kararların hızlı yayılması
- Veri bağımlılığı
- Model hatalarının operasyonlara etkisi

Bu nedenle AIP gibi sistemlerde güçlü kontrol katmanları bulunur. İnsan onayı, güvenlik sınırları ve şeffaflık mekanizmaları kritik rol oynar.

Palantir Technologies tarafından geliştirilen AIP yaklaşımı, yapay zekânın geleceğini temsil eden önemli bir dönüşüm noktasıdır. LLM entegrasyonu, bilgi grafı tabanlı bağlam ve operasyonel ajan sistemleri birleşerek yapay zekâyı yalnızca bir analiz aracı olmaktan çıkarır ve onu doğrudan operasyonel karar üretim mekanizmasına dönüştürür.

Bu yeni paradigma, veri çağından "karar çağ"ına geçişi temsil eder. Gelecekte rekabet avantajı yalnızca daha fazla veri sahibi olmakla değil, veriyi gerçek zamanlı olarak eyleme dönüştürebilme kapasitesiyle belirlenecektir.

11.9 Project Maven ve Askeri YZ Ekosistemi

21.yüzyılın ikinci çeyreğine girildiğinde yapay zekâ artık yalnızca ticari yazılımların ya da akademik laboratuvarların konusu olmaktan çıkmıştı. Veri, algoritma ve hesaplama gücünün birleşimi, özellikle devletlerin güvenlik ve istihbarat kapasitesini yeniden tanımlayan bir eksene dönüşmüştü. Bu dönüşümün en kritik eşiklerinden biri, ABD Savunma Bakanlığı tarafından başlatılan Project Maven oldu. Maven, yalnızca bir teknoloji projesi değil, aynı zamanda modern savaş alanında "algı" kavramının nasıl otomatikleştiğini gösteren tarihsel bir kırılma noktasıydı.

Project Maven'in temel problemi oldukça basitti, ancak çözümü son derece derindi: İnsan gözünün işleyemeyeceği kadar büyük bir görüntü akışını anlamlandırmak. Uydu görüntüleri, drone videoları ve sensör verileri her gün petabaytlarca veri üretmekteydi. Geleneksel analiz yöntemleri bu veri yığınını işleyemiyor, insan analistlerin dikkat kapasitesi ciddi bir darboğaz oluşturuyordu. Maven bu noktada devreye girerek görüntü tanıma ve nesne tespiti için derin öğrenme modellerini askeri istihbarat süreçlerine entegre etti.

Bu teknik dönüşümün özünde aslında basit ama güçlü bir fikir vardı: Görmek artık insanın tekeline ait olmayacaktı. Yapay zekâ, görüntü içindeki nesneleri sınıflandıracak, hareketleri izleyecek ve potansiyel tehditleri işaretleyecekti. Bu, askeri istihbaratın "yorumlama" katmanını yeniden tasarlayan bir adımdı. Çünkü artık veri sadece toplanmıyor, aynı zamanda otomatik olarak anlamlandırılıyordu.

Project Maven'in Palantir ekosistemiyle ilişkisi bu noktada kritik hale gelir. Palantir'in Gotham platformu istihbarat verisini ilişkisel bir ağ olarak işlerken, Maven bu ağın en alt katmanına, yani "algı katmanına" yerleşiyordu. Başka bir deyişle Gotham, dünyayı bir graf yapısı olarak temsil ederken; Maven bu grafiğin ham girdisini üretiyordu: görüntüden çıkarılmış varlıklar (entities), konumlar, hareketler ve nesneler.

Bu dönüşüm, "operasyonel bilgi grafı" kavramının üç temel bileşenini ortaya koydu:

Birincisi, veri toplama katmanıydı. Uydu ve drone sistemleri sürekli bir veri akışı sağlıyordu. İkincisi, yapay zekâ katmanıydı; burada görüntü tanıma algoritmaları nesneleri tespit ediyor ve etiketliyordu. Üçüncüsü ise operasyon katmanıydı; burada insan analistler ya da otomatik sistemler bu bilgiyi karar mekanizmalarına dönüştürüyordu.

Bu zincir, klasik istihbarat modelinden radikal biçimde farklıydı. Eskiden bir görüntü analisti bir fotoğrafı inceler, şüpheli bir nesneyi işaretler ve rapor yazardı. Maven sonrası sistemde ise bu süreç büyük ölçüde otomatikleşti. İnsan artık "gören" değil, "doğrulayan" pozisyona geçti.

Ancak Project Maven yalnızca teknik bir devrim değildi; aynı zamanda Silikon Vadisi ile askeri-endüstriyel kompleks arasındaki sınırın bulanıklaşmasını simgeliyordu. Google'ın projeye dahil olması, teknoloji şirketleri ile savunma sanayi arasındaki etik ve politik gerilimleri görünür hale getirdi. Şirket içi protestolar ve kamuoyu tartışmaları, yapay zekânın askeri kullanımına dair yeni bir toplumsal bilinç doğurdu.

Bu durum, daha geniş bir "askeri YZ ekosistemi"nin oluşumuna işaret eder. Maven tek başına bir proje değil, bir ağın başlangıç düğümüdür. Bu ağ içinde veri sağlayıcı uydu sistemleri, otonom drone filoları, görüntü işleme modelleri, karar destek platformları ve operasyon merkezleri birbiriyle entegre çalışır. Bu yapı, klasik komuta kontrol sistemlerinin yerini alan dağıtık bir zekâ mimarisidir.

Palantir bu ekosistemin en önemli yazılım omurgalarından birini sağlar. Gotham, farklı kaynaklardan gelen verileri birleştirerek ilişkisel bir anlam haritası oluşturur. Foundry ise benzer yaklaşımı sivil ve kurumsal dünyaya taşır. Maven ise bu sistemin "algısal giriş kapısı"dır. Bu üçlü birlikte düşünüldüğünde, modern devletin dijital sinir sistemi ortaya çıkar: algı (Maven), anlam (Gotham/Foundry) ve karar (operasyonel sistemler).

Bu ekosistemin en dikkat çekici yönlerinden biri hızdır. Geleneksel istihbarat döngüsü günler, haftalar hatta aylar alabilirken, YZ destekli sistemler saniyeler içinde analiz üretebilmektedir. Bu hız farkı, yalnızca teknik bir avantaj değil, aynı zamanda stratejik bir güç çarpanı yaratır. Çünkü modern çatışma ortamında "ilk gören, ilk karar veren" taraf avantaj sağlar.

Ancak bu hızın beraberinde getirdiği ciddi bir problem vardır: yorumlama hatalarının ölçeklenmesi. Yapay zekâ sistemleri yanlış bir nesne tespit ettiğinde, bu hata yalnızca bir analistin raporunda kalmaz; tüm operasyonel zincire yayılabilir. Bu nedenle insan denetimi tamamen ortadan kalkmaz, ancak rolü dönüşür. İnsan artık veri üreticisi değil, sistem denetleyicisidir.

Askeri YZ ekosisteminin bir diğer boyutu da otonom sistemlere doğru evrimdir. Drone sürüleri, otonom keşif araçları ve yapay zekâ destekli komuta sistemleri, Maven gibi projelerin açtığı kapıdan geçerek daha karmaşık yapılar oluşturur. Bu yapı, savaşın doğasını değiştirerek onu giderek daha "hesaplanabilir" bir problem haline getirir. Ancak aynı zamanda savaşın etik boyutunu daha da karmaşıklaştırır.

Aslında **Project Maven** adıyla başlayan bu girişim, yapay zekânın savaş alanında karar destek mekanizması olarak nasıl devreye girdiğinin en somut örneği.

- **Ne İşe Yarar?** Maven, insansız hava araçları (İHA), uydular, sensörler ve radarlardan gelen ham veri akışını anında işleyen bir makine öğrenimi platformudur. Geleneksel

yöntemlerle analistlerin saatler veya günler süren işlemlerini, Maven dakikalar hatta saniyeler içinde tamamlayabiliyor.

- **Savaş Planlaması ve Yapay Zekâ Asistanları:** Palantir'in yapay zekâ platformu (AIP) üzerinden çalışan bu sistem, askeri personele adeta bir savaş asistanı gibi hizmet veriyor. Örneğin, WIRED dergisinin incelediği demo videolarında, bir analistin sohbet robotuna yazdığı "**Bu bölgedeki düşman askeri birliği hangisi?** " sorusuna sistem, radar görüntülerini analiz ederek "**Zırhlı bir tabur olabilir**" cevabını verebiliyor.
- **Operasyonel Yetenekler:** Analist daha sonra aynı sohbet robotuna "**Bu düşman teçhizatını hedef almak için üç eylem planı oluştur**" komutunu yazabiliyor. Yapay zekâ ise saniyeler içinde "Hava saldırısı", "Uzun menzilli topçu" veya "Taktik tim" gibi seçenekler sunuyor. Sistem ayrıca birlikler için rota oluşturma, düşman iletişimine karışma ve lojistik takibi gibi karmaşık görevleri de üstlenebiliyor.

Sonuç olarak Project Maven, yalnızca bir görüntü tanıma projesi değildir. O, modern askeri yapının veri, algoritma ve karar süreçleriyle yeniden inşa edilmesinin başlangıç noktasıdır. Palantir gibi platformlarla birleştiğinde ise ortaya çıkan şey, yalnızca bir yazılım sistemi değil, dünyanın gerçek zamanlı olarak modellenmesi fikridir. Bu model, hem olağanüstü bir operasyonel güç üretir hem de yeni bir gözetim ve güç yoğunlaşması çağını başlatır.

Askeri YZ Sistemlerinin Yazılım Anatomisi

Modern askeri yapay zekâ sistemleri, tek bir programdan oluşan klasik yazılımlar değildir. Bunlar; veri toplama, görüntü işleme, makine öğrenmesi, karar desteği ve operasyon yönetimini aynı anda yürüten çok katmanlı dijital organizmalardır. Bu nedenle askeri YZ mimarileri, farklı görevler için farklı programlama dilleri ve teknolojik katmanlar kullanır.

Örneğin Python, derin öğrenme modellerinin geliştirilmesinde merkezi rol oynar. Özellikle PyTorch ve TensorFlow gibi araçlarla görüntü tanıma sistemleri eğitilir. CUDA tabanlı GPU sistemleri ise bu modellerin milyonlarca görüntüyü yüksek hızda işlemesini sağlar. Performans kritik alanlarda C++ kullanılarak gerçek zamanlı analiz mümkün hale getirilir.

Görsel verinin ön işlenmesinde OpenCV önemli bir rol oynarken, Kubernetes dağıtık YZ servislerinin yönetimini üstlenir. Modellerin farklı sistemler arasında taşınabilmesi için ise ONNX kullanılır. Sonuçta ortaya çıkan yapı, yalnızca veri işleyen bir yazılım değil; dünyayı algılayan, yorumlayan ve operasyonel kararlara dönüştüren dijital bir sinir sistemidir.

Gotham ve Foundry’nin Yazılım Anatomisi: Çok Dilli Bir Operasyonel Zekâ

Palantir ekosistemini anlamanın en zor yanı, onun tek bir yazılım ürünü gibi düşünülmesidir. Oysa Palantir Gotham ve Palantir Foundry, klasik anlamda monolitik yazılımlar değildir. Bunlar, farklı veri türlerini, farklı kullanıcı rollerini ve farklı karar katmanlarını aynı sistem içinde birleştiren çok katmanlı (multi-layer) ve çok dilli (polyglot) bir mimari üzerine kuruludur.

Bu mimarinin temel fikri şudur: Tek bir programlama dili, gerçek dünyanın karmaşıklığını temsil etmeye yetmez. Çünkü gerçek dünya; veri tabanları, akış sistemleri, kullanıcı arayüzleri, makine öğrenmesi modelleri ve ontolojik ilişkiler gibi birbirinden farklı bilişsel katmanlardan oluşur. Bu nedenle Gotham ve Foundry, her katman için farklı bir "dil ailesi" kullanır. Bu yapı, yalnızca teknik bir tercih değildir; aslında bir tasarım felsefesidir. Sistem, veriyi sadece saklamaz; onu farklı "anlama biçimlerine" dönüştürür.

Önce iki örnek verelim:

```
# Gotham - entity linking example

person = graph.get_entity("Person:123")
associates = graph.traverse(person, relation="communicated_with")

risk_score = 0
for node in associates:
   risk_score += node.weight * node.threat_level
if risk_score > THRESHOLD:
   alert("high_risk_network", person)


# Foundry - data pipeline example

sales = load_dataset("sales_db")
logistics = load_dataset("shipping_api")
joined = join(sales, logistics, on="order_id")
cleaned = transform(joined, rules="standardize_dates")
publish_to_graph(cleaned, ontology="enterprise_graph")
```

Backend Katmanı: Sistemin Sinir Sistemi (Java ve C++)

Backend, Gotham ve Foundry'nin omurgasıdır. Tüm veri akışı, işlem mantığı, erişim kontrolü ve graf yapıların yönetimi bu katmanda gerçekleşir.

Burada ağırlıklı olarak:

- Java
- C++

kullanılır.

Java, sistemin kurumsal ve ölçeklenebilir kısmını taşır. Büyük veri akışlarının yönetimi, servisler arası iletişim ve dağıtık sistemlerin koordinasyonu Java ekosistemi üzerinden yürütülür. Java'nın burada seçilmesinin nedeni yalnızca performans değil, aynı zamanda güvenilirlik ve uzun ömürlü sistem mimarisi sağlamasıdır.

C++ ise sistemin daha "altyapı" düzeyindeki işlerini üstlenir. Özellikle hızın ve performansın kritik olduğu durumlarda tercih edilir. Örneğin çok büyük veri graflarında bağlantıların hızlı

biçimde taranması (graph traversal), gerçek zamanlı analiz sistemleri ve milisaniyeler düzeyinde çalışan düşük gecikmeli hesaplama modülleri genellikle C++ ile geliştirilir. Bu nedenle C++, yapay zekâ ve veri sistemlerinin "motor kısmında" çalışan temel teknolojilerden biri olarak görülür.

Basitçe söylemek gerekirse:

- Java → sistemin "organizasyon zekâsı"
- C++ → sistemin "refleksleri"

Backend katmanı, kullanıcıya görünmezdir. Ancak sistemin tüm ağırlığı burada taşınır. Bu katman, veri grafının sürekli yaşayan bir organizma gibi çalışmasını sağlar.

Veri Katmanı: SQL ve Yapısal Gerçeklik

Gotham ve Foundry'nin en temel bileşenlerinden biri veri katmanıdır. Burada klasik anlamda:

- SQL
- ilişkisel veri modelleri
- veri ambarları

kullanılır.

SQL, sistemin "düzen dili"dir. Veriyi satırlar ve sütunlar halinde tanımlayan klasik yaklaşım, Palantir mimarisinde tamamen ortadan kalkmaz; aksine daha büyük bir yapının alt bileşeni haline gelir. Ancak önemli bir dönüşüm vardır: SQL artık nihai yapı değil, graf modelinin besleyici kaynağıdır. Yani veri artık sadece tablolar içinde durmaz; daha sonra ilişkilere dönüştürülür, düğümlere ve bağlantılara evrilir. Bu dönüşüm, klasik veri mühendisliğinden farklı olarak şunu ifade eder: Veri, artık bir kayıt değil; bir ilişkiler ağının hammaddesidir.

Analitik Katman: Python ve Makine Öğrenmesi

Sistemin "akıl yürütme" katmanı Python merkezlidir. Özellikle:

- veri analizi
- makine öğrenmesi
- istatistiksel modelleme
- YZ entegrasyonu

Python üzerinden yürütülür.

Bu katmanda Palantir Foundry içinde çalışan analitik pipeline'lar veya Palantir Gotham üzerindeki istihbarat modelleri, Python tabanlı algoritmalarla beslenir.

Python burada yalnızca bir programlama dili değildir; yorumlayıcı bir düşünme ortamıdır. Veri, Python içinde işlenir, temizlenir, modelden geçirilir ve tekrar graf yapısına geri döner.

Basit bir bakışla:

- SQL → veri
- Python → bilgi üretimi
- Graf sistemi → anlam

UI Katmanı: TypeScript / JavaScript ve Görsel Zekâ

Kullanıcıların sistemi "gördüğü" yer UI katmanıdır. Burada:

- TypeScript
- JavaScript
- modern frontend framework'ler

kullanılır.

UI (User Interface), sadece görsellik değildir. Bu sistemlerde UI, veri grafının insan tarafından okunabilir versiyonudur. Kullanıcı, karmaşık ilişkisel ağları doğrudan backend'deki haliyle değil, görselleştirilmiş bir "karar alanı" olarak görür.

Bu nedenle UI katmanı şu işlevleri yerine getirir:

- Graf yapıları görselleştirme
- İlişkileri interaktif olarak gösterme
- Zaman serilerini ve hareketleri animasyonlaştırma
- Kullanıcının veriye müdahale etmesini sağlama

TypeScript'in burada kullanılması, sistemin büyüklüğünden kaynaklanır. Büyük veri görselleştirme sistemlerinde tip güvenliği kritik hale gelir.

Bu katman aslında şunu yapar:

Veriyi insan gözünün anlayabileceği bir bilişsel haritaya dönüştürür.

Ontoloji Katmanı: Sistemin Kendi Dili (DomYZn DSL)

Gotham ve Foundry'nin en özgün katmanı, klasik programlama dillerinden farklıdır. Bu katmanda Palantir'in kendi geliştirdiği:

- ontology (ontoloji modeli)
- schema tanımlama dili (DSL)

kullanılır.

Bu katman, sistemin “ne olduğunu” tanımlar. Örneğin:

- Bir “kişi” nedir?
- Bir “varlık (asset)” nedir?
- Bir “risk” nasıl tanımlanır?
- Bir “ilişki” hangi türden olabilir?

Bu DSL, aslında sistemin anlam sözlüğüdür.

Bu nedenle en kritik katmanlardan biridir. Çünkü burada yapılan hata, tüm graf yapısını yanlış yorumlamaya neden olur.

Ontoloji katmanı olmadan sistem sadece veri yığınıdır. Ontoloji ile birlikte sistem bir anlam makinesi haline gelir.

Çok Dilli Mimari: Sistemlerin Bilişsel Haritası

Tüm bu yapı bir araya geldiğinde ortaya çıkan şey şudur:

Gotham ve Foundry tek bir dil ile yazılmış sistemler değildir. Bunlar, farklı veri gerçekliklerini bir araya getirmek için çok dilli (polyglot) bir mimari kullanır. Her dil, sistemin farklı bir bilişsel katmanına karşılık gelir.

Bu katmanlar birlikte çalıştığında ortaya çıkan yapı şudur:

- Java/C++ → sistemin fiziksel motoru
- SQL → veri hafızası
- Python → düşünme ve analiz katmanı
- TypeScript/JavaScript → görsel bilinç
- Ontoloji DSL → anlam sistemi

Bu mimari, klasik yazılım mühendisliğinden farklı olarak bir “bilgi sistemi” değil, bir karar üretim organizmasıdır. Bu teknik yapı yalnızca mühendislik değildir. Palantir Gotham ve Palantir Foundry, yazılım katmanlarını aşarak gerçek dünyanın operasyonel temsilcisine dönüşür. Backend veri işler, Python anlam üretir, UI bunu görünür kılar, ontoloji ise tüm sistemi “ne olduğuna” dYZr bir çerçeve içine alır. Sonuçta ortaya çıkan şey bir yazılım değil, giderek daha fazla şeye benzeyen bir yapıdır:

Gerçek dünyanın dijital bir sinir sistemi.

Project Maven’in Teknik Anatomisi: YZ/Vision Stack ve Algısal Savaş Mimarisi

Project Maven modern askeri yapay zekâ sistemlerinin en önemli dönüm noktalarından biridir. Çünkü Maven yalnızca bir görüntü tanıma sistemi değil, aynı zamanda “algısal otomasyon”

mimarisidir. Bu sistemin temel amacı, insan gözünün yetişemeyeceği büyüklükteki görüntü akışını gerçek zamanlı olarak analiz etmek ve operasyonel anlam üretmektir.

Uydu sistemleri, keşif uçakları, insansız hava araçları (drone), termal kameralar ve radarlar sürekli veri üretir. Modern savaş alanı artık yalnızca fiziksel değil; aynı zamanda devasa bir veri ortamıdır. Project Maven'in ortaya çıkış nedeni tam da budur: İnsan analistler bu veri hacmini artık işleyememektedir.

Bu nedenle Maven, klasik istihbarat zincirini yeniden yapılandırır:

- Görüntü toplama
- Görüntüyü anlamlandırma
- Nesne tanıma
- Davranış analizi
- Risk tahmini
- Operasyonel karar desteği

Bu zincirin arkasında ise çok katmanlı bir YZ/Vision Stack bulunur.

Python: Yapay Zekânın Operasyonel Dili

Maven mimarisinin merkezinde Python bulunur. Günümüzde yapay zekâ ekosisteminin ortak dili haline gelen Python, özellikle:

- derin öğrenme,
- veri analizi,
- görüntü işleme,
- model eğitimi,
- istatistiksel analiz

alanlarında temel araçtır.

Project Maven gibi sistemlerde Python'un bu kadar güçlü olmasının nedeni, yalnızca dilin basitliği değildir. Asıl neden, Python'un devasa YZ ekosistemidir.

Özellikle:

- PyTorch
- TensorFlow

gibi framework'ler sayesinde görüntü tanıma modelleri hızlı biçimde geliştirilebilir.

Bu modeller:

- tank,
- araç,
- insan,
- bina,
- silah sistemi,
- hareket paterni

gibi nesneleri tanımayı öğrenir.

Örneğin bir drone görüntüsü sisteme geldiğinde, Python tabanlı model şu tür işlemler yapabilir:

```
image = load_drone_frame()

objects = model.detect(image)

for obj in objects:
   if obj.label == "vehicle":
      track(obj)
```

Bu örnek basit görünse de arkasında milyonlarca parametre içeren sinir ağları bulunur. Sistem artık görüntüyü "görmez"; onu matematiksel özellik uzaylarına dönüştürür.

Bu nedenle Python burada yalnızca bir yazılım dili değildir. O, modern savaş alanının yorumlayıcı katmanıdır.

CUDA: GPU Gücü ve Paralel Hesaplama

Ancak modern YZ sistemlerinin asıl gücü yalnızca algoritmalardan gelmez. Büyük dönüşüm, GPU tabanlı hesaplamanın ortaya çıkışıyla başlamıştır.

Burada kritik teknoloji:

- CUDA'dır.

CUDA

CUDA, NVIDIA tarafından geliştirilen paralel hesaplama mimarisidir. Geleneksel işlemciler (CPU) sıralı işlem yaparken GPU'lar binlerce işlemi aynı anda gerçekleştirebilir.

Bu fark özellikle görüntü işleme için kritik hale gelir.

Çünkü bir uydu görüntüsü milyonlarca piksel içerir. Derin öğrenme modelleri her piksel üzerinde matematiksel işlemler yapar. CPU ile bu süreç çok yavaş olurdu.

CUDA sayesinde:

- binlerce çekirdek aynı anda çalışır,
- sinir ağları hızlanır,
- gerçek zamanlı analiz mümkün hale gelir.

Askeri açıdan bunun anlamı şudur:

Görüntü analizi artık saatler değil, saniyeler içinde yapılabilir.

Bu hız, modern savaşta doğrudan stratejik üstünlük anlamına gelir.

C++: Gerçek Zamanlı Performans Katmanı

Project Maven'de Python geliştirme dili olarak çok güçlüdür; ancak operasyonel sistemlerde tek başına yeterli değildir.

Çünkü savaş alanında milisaniyeler önemlidir.

Bu nedenle performans kritik bölümlerde:

- C++

kullanılır.

Özellikle:

- model inference optimization,
- düşük gecikmeli analiz,
- yüksek hızlı veri akışı,
- gerçek zamanlı karar sistemleri

C++ tabanlı optimize modüllerle çalışır.

Burada önemli bir kavram vardır: inference.

Model eğitmek başka, modeli sahada çalıştırmak başkadır.

Örneğin milyonlarca görüntü üzerinde eğitilmiş bir modelin gerçek operasyon sırasında:

- hızlı,
- kararlı,
- düşük enerji tüketimli

çalışması gerekir.

İşte bu nedenle Python ile geliştirilen birçok model, daha sonra C++ tabanlı yüksek performans motorlarına dönüştürülür. Bu yapı, YZ sistemlerinin "araştırma laboratuvarı"ndan çıkıp gerçek dünyaya geçmesini sağlar.

OpenCV: Görsel Dünyanın Ön İşleme Motoru

Maven mimarisinin önemli parçalarından biri de:

- OpenCV

kütüphanesidir.

OpenCV, bilgisayarlı görü (computer vision) sistemlerinin temel araçlarından biridir.

Derin öğrenme modelleri doğrudan ham görüntüyle çalışmaz. Önce görüntünün işlenmesi gerekir:

- gürültü azaltma,
- kontrast düzenleme,
- kenar tespiti,
- hareket analizi,
- nesne segmentasyonu

gibi işlemler yapılır.

OpenCV bu "ön algı" katmanını oluşturur.

İnsan beyninde göz nasıl görsel bilgiyi ön işleme tabi tutuyorsa, OpenCV de dijital sistemlerde benzer bir işlev görür.

Bu nedenle OpenCV:

Yapay zekânın dijital retinası olarak düşünülebilir.

Kubernetes: YZ Sistemlerinin Operasyonel Orkestrasyonu

Project Maven yalnızca tek bir bilgisayarda çalışan model değildir. Bu sistemler:

- veri merkezleri,
- bulut altyapıları,
- askeri ağlar,
- edge cihazlar,
- drone sistemleri

üzerinde dağıtık biçimde çalışır.

Bu nedenle sistemin yönetimi için:

- Kubernetes

kullanılır.

Kubernetes'in temel görevi:

- YZ servislerini dağıtmak,
- ölçeklemek,
- güncellemek,
- hata durumunda yeniden başlatmak

ve sistemi sürekli çalışır halde tutmaktır.

Modern askeri YZ sistemleri artık yalnızca algoritma değildir; yaşayan bir dijital organizmadır.

Kubernetes bu organizmanın dolaşım sistemidir.

Örneğin savaş alanındaki farklı drone sistemleri aynı YZ modelinin farklı sürümlerini çalıştırabilir. Kubernetes bu dağıtımı merkezi biçimde yönetir.

ONNX: Modellerin Ortak Dili

YZ ekosisteminin en büyük problemlerinden biri model taşınabilirliğidir.

Bir model:

- PyTorch'ta eğitilmiş olabilir,
- TensorFlow'da çalıştırılabilir,
- edge cihazda deploy edilebilir.

Bu geçişleri mümkün kılan yapı:

- ONNX

sistemidir.

ONNX, farklı YZ framework'leri arasında ortak bir köprü görevi görür.

Bu askeri sistemlerde çok önemlidir çünkü:

- farklı donanımlar,
- farklı GPU'lar,
- farklı operasyonel ortamlar

aynı modeli kullanmak zorunda olabilir.

ONNX sayesinde model bir platformdan diğerine taşınabilir.

Bu yapı aslında YZ dünyasının "evrensel ara dili"dir.

Sonuç: Algısal Savaşın Yazılım Mimarisi

Project Maven'in teknik stack'i yalnızca mühendislik tercihlerinden oluşmaz. Her katman, modern savaşın farklı bir bilişsel işlevine karşılık gelir:

- Python → öğrenme ve yorumlama
- CUDA → hız ve paralel algı
- C++ → refleks ve performans
- OpenCV → görsel ön algı
- Kubernetes → operasyonel koordinasyon
- ONNX → sistemler arası taşınabilirlik

Bu yapı birlikte çalıştığında ortaya çıkan şey klasik anlamda bir yazılım sistemi değildir.Ortaya çıkan şey şudur: Dünyayı gerçek zamanlı algılayan, yorumlayan ve operasyonel karara dönüştüren dijital bir sinir sistemi.

11.10 Büyük Güç Mü, Büyük Gözetim Mi?

Bilgi grafı, yapay zekâ ve operasyonel veri sistemlerinin yükselişi yalnızca teknolojik bir dönüşüm değildir; aynı zamanda derin bir politik ve felsefi kırılmadır. Çünkü veri artık yalnızca teknik bir kaynak değil, aynı zamanda güç üretiminin temel maddesi haline gelmiştir. Bu nedenle modern dünyada giderek daha sık sorulan bir soru vardır: Bu yeni sistemler insanlığa "büyük bir güç" mü sağlıyor, yoksa "büyük bir gözetim" mekanizmasına mı dönüşüyor?

Bu tartışma özellikle Palantir Technologies gibi operasyonel bilgi grafı sistemleri geliştiren şirketlerin yükselişiyle daha da görünür hale gelmiştir. Çünkü bu tür platformlar, farklı veri kaynaklarını birleştirerek bireyler, kurumlar ve toplumlar hakkında son derece ayrıntılı ilişki haritaları oluşturabilir.

Bu durum teknolojinin doğasından kaynaklanır: Bir sistemi ne kadar iyi ilişkilendirebilirseniz, o sistemi o kadar iyi kontrol edebilirsiniz.

Mahremiyet tartışmaları: görünmeyen sınırların erimesi

Dijital çağın en temel sorunlarından biri mahremiyetin yeniden tanımlanmasıdır. Klasik dünyada mahremiyet, fiziksel sınırlar ve bireysel alanlarla korunurdu. Ancak dijital dünyada mahremiyet artık veri akışları üzerinden belirlenmektedir.

Bilgi grafı sistemleri:

- İnsan ilişkilerini
- Hareket verilerini
- Finansal işlemleri
- İletişim ağlarını

- Dijital davranış örüntülerini

birbirine bağlayarak bütünsel bir model oluşturur.

Bu model teknik açıdan son derece güçlüdür; ancak aynı zamanda mahremiyet açısından yeni riskler yaratır. Çünkü bireyler artık tekil veri noktaları olarak değil, sürekli genişleyen ilişki ağlarının parçası olarak temsil edilir.

Bu durum şu soruyu gündeme getirir:

Bir bireyin davranışı ne kadar "kişisel" kalabilir?

Eğer tüm ilişkiler analiz edilebiliyorsa, anonimlik kavramı giderek zayıflar.

Devlet–şirket ilişkileri: yeni güç mimarisi

Bilgi grafı ve yapay zekâ sistemlerinin yükselişi, devletler ile teknoloji şirketleri arasındaki ilişkiyi kökten değiştirmiştir. Eskiden devletler veri toplama ve analiz konusunda açık ara baskındı. Ancak günümüzde büyük teknoloji şirketleri çok daha gelişmiş veri işleme altyapılarına sahiptir.

Bu durum yeni bir güç dengesi yaratmıştır:

- Devletler düzenleyici güçtür
- Şirketler ise veri altyapısının sahibidir

Bu ikili yapı zaman zaman iş birliği, zaman zaman gerilim üretir.

Operasyonel bilgi grafı sistemleri bu ilişkinin merkezindedir. Çünkü bu sistemler:

- Kamu güvenliği
- Sağlık yönetimi
- Ekonomik analiz
- Altyapı kontrolü

gibi alanlarda kullanılabilir.

Bu nedenle devlet ile şirket arasındaki sınır giderek bulanıklaşmaktadır. Hangi verinin kime YZt olduğu, hangi analizlerin kim tarafından yapılabileceği ve hangi kararların kim tarafından verileceği giderek daha karmaşık hale gelmektedir.

Veri egemenliği: dijital çağın yeni bağımsızlık sorunu

Geleneksel dünyada egemenlik, toprak ve kaynak kontrolüyle tanımlanıyordu. Ancak dijital çağda egemenlik artık veri kontrolü ile ölçülmektedir.

"Veri egemenliği" kavramı şu soruyu merkezine alır:

Bir toplum kendi verisini kim yönetiyor?

Eğer bir ülkenin vatandaşlarına YZt veriler, yabancı şirketlerin altyapılarında işleniyorsa, bu durum sadece teknik bir tercih değildir; aynı zamanda politik bir bağımlılık ilişkisidir.

Bilgi grafı sistemleri bu problemi daha da derinleştirir. Çünkü:

- Veriler yalnızca depolanmaz
- Aynı zamanda ilişkilendirilir
- Davranış modelleri çıkarılır
- Gelecek tahminleri yapılır

Bu nedenle veri yalnızca geçmişi değil, geleceği de temsil eden bir güç haline gelir.

Veri egemenliği tartışması bu yüzden modern çağın en kritik jeopolitik meselelerinden biridir.

Dijital Leviathan korkusu

Tarih boyunca güç yoğunlaşması her zaman yeni metaforlarla anlatılmıştır. Thomas Hobbes'un "Leviathan" kavramı, merkezi ve devasa devlet gücünü temsil eder. Dijital çağda ise bu kavram yeni bir biçim kazanmıştır: "Dijital Leviathan".

Dijital Leviathan, tüm veri akışlarını görebilen, ilişkilendirebilen ve analiz edebilen merkezi bir sistem metaforudur. Bu sistem:

- Her bireyin dijital izini takip edebilir
- Sosyal ağları analiz edebilir
- Ekonomik davranışları modelleyebilir
- Operasyonel kararlar üretebilir

Bu nedenle hem büyük bir verimlilik hem de büyük bir kontrol gücü anlamına gelir.

Palantir Technologies gibi platformlar, bu tür sistemlerin teknik olarak mümkün olduğunu göstermiştir. Ancak bu durum aynı zamanda önemli bir korkuyu da beraberinde getirir: Eğer tüm bilgi tek bir sistemde yoğunlaşırsa, güç de aynı oranda merkezileşir.

Bu korku üç temel soruda toplanır:

- Kim sistemi kontrol ediyor?
- Sistem hangi değerlerle çalışıyor?
- Hatalar nasıl engelleniyor?

Güç ve gözetim arasındaki ince çizgi

Aslında modern veri sistemlerinin en büyük paradoksu şudur: Aynı teknoloji hem özgürleştirici hem de kontrol edici olabilir.

- Daha iyi veri analizi → daha iyi sağlık sistemleri
- Daha iyi lojistik → daha verimli ekonomi
- Daha iyi güvenlik analizi → daha güvenli toplum

Ancak aynı sistemler:

- Aşırı gözetim
- Davranış manipülasyonu
- Otomatik karar baskısı
- Algoritmik kontrol

risklerini de taşır.

Bu nedenle mesele teknolojik değil, yapısaldır.

Bilgi grafı sistemleri nötr değildir; çünkü hangi verilerin nasıl ilişkilendirileceğine dYZr seçimler içerir. Bu seçimler ise güç üretir.

İnsan faktörü ve denge arayışı

Tüm bu tartışmaların merkezinde insan faktörü yer alır. Teknoloji ne kadar gelişirse gelişsin, karar mekanizmalarının etik ve politik sorumluluğu insana YZttir.

Bu nedenle gelecekte en önemli meselelerden biri şudur:

- Sistemler ne kadar otomatikleşmeli?
- İnsan ne kadar kontrol etmeli?
- Şeffaflık nasıl sağlanmalı?
- Güç nasıl dengelenmeli?

Bu sorulara verilecek cevaplar, dijital çağın yönünü belirleyecektir.

Büyük veri, bilgi grafı ve yapay zekâ sistemleri insanlığa benzeri görülmemiş bir güç kazandırmaktadır. Ancak bu güç aynı zamanda yeni bir gözetim mimarisi oluşturma potansiyeli de taşır. Palantir Technologies gibi sistemler bu ikili doğanın en somut örneklerindendir: bir yandan operasyonel zekâ üretirken, diğer yandan geniş ölçekli veri görünürlüğü sağlar.

Sonuç olarak modern çağın temel sorusu teknik değil, felsefidir: Bu yeni dijital Leviathan insanlığın hizmetinde bir araç mı olacak, yoksa insan davranışlarını yöneten görünmez bir yapı mı haline gelecek?

Cevap, teknolojiden çok onu nasıl tasarladığımıza ve nasıl yönettiğimize bağlıdır.

EKLER

Ek A — Temel Graf Kuramı

Graf kuramı, modern veri bilimi, bilgisayar bilimi, ağ analitiği ve bilgi grafı sistemlerinin temel matematiksel omurgasını oluşturur. En basit tanımıyla bir graf, düğümler (nodes/vertices) ve bu düğümleri birbirine bağlayan bağlantılardan (edges/links) oluşan bir yapıdır. Ancak bu basit tanımın arkasında, son derece zengin ve çok katmanlı bir matematiksel dünya bulunur.

Bir graf genellikle şu şekilde tanımlanır: **G = (V, E)** Burada V, düğüm kümesini; E ise düğümler arasındaki bağlantı kümesini temsil eder. Eğer bir bağlantı yönlü ise (örneğin A'dan B'ye), buna yönlü graf (directed graf) denir. Eğer bağlantılar çift yönlü ise yönsüz graf söz konusudur.

Graf kuramı, yalnızca soyut bir matematiksel yapı değildir; gerçek dünyadaki birçok sistemin doğal modelidir. Sosyal ağlar, ulaşım sistemleri, biyolojik etkileşimler, internet altyapısı ve bilgi grafı sistemleri doğrudan graf yapılarıyla temsil edilir.

Düğüm ve Bağlantı Kavramı

Düğüm, bir varlığı temsil eder. Bu bir kişi, bir şehir, bir makine, bir belge ya da bir kavram olabilir. Bağlantı ise bu varlıklar arasındaki ilişkiyi ifade eder. Örneğin:

- Bir kişi "çalışır" bir şirkette
- Bir şehir "bağlıdır" bir ulaşım ağına
- Bir protein "etkileşir" başka bir proteinle

Bu ilişkiler graf yapısında bağlantılarla gösterilir.

Bağlantılar ayrıca ağırlıklı olabilir. Bu durumda ilişkilerin gücü, maliyeti veya mesafesi gibi ek bilgiler taşınır.

Komşuluk ve Derece Kavramı

Bir düğümün "komşuları", doğrudan bağlı olduğu düğümlerdir. Bir düğümün derece değeri, o düğüme bağlı bağlantı sayısını ifade eder. Yönlü graflarda giriş derecesi (in-degree) ve çıkış derecesi (out-degree) olmak üzere iki farklı ölçüm vardır. Bu kavramlar, özellikle sosyal ağ analizi ve bilgi grafı sistemlerinde kritik öneme sahiptir. Çünkü yüksek dereceli düğümler genellikle merkezi veya etkili varlıkları temsil eder.

Yol (Path) ve Bağlantılılık

Bir graf içinde bir düğümden diğerine ulaşmak için izlenen düğüm dizisine yol (path) denir. Eğer bir graf içindeki her düğüm diğer düğümlere bir yol ile ulaşabiliyorsa, bu graf bağlantılı (connected) olarak adlandırılır.

Bu kavram özellikle ağ dayanıklılığı açısından önemlidir. Örneğin bir iletişim ağında bağlantılılık, sistemin kesintilere karşı ne kadar dayanıklı olduğunu belirler.

Döngüler ve Ağaç Yapıları

Bir graf içinde başlangıç ve bitiş düğümünün aynı olduğu yollar döngü (cycle) olarak adlandırılır. Döngü içermeyen bağlantılı graflar ise ağaç (tree) yapısını oluşturur.

Ağaç yapıları, özellikle hiyerarşik veri modellemede kullanılır. Dosya sistemleri, organizasyon şemaları ve karar ağaçları bu yapıya örnektir.

Merkeziyet Ölçümleri

Graf kuramının en önemli uygulama alanlarından biri merkeziyet analizidir. Bir düğümün graf içindeki "önemini" ölçmek için çeşitli metrikler kullanılır:

- Degree centrality: Bağlantı sayısı
- Betweenness centrality: Ara düğüm olma durumu
- Closeness centrality: Diğer düğümlere uzaklık
- Eigenvector centrality: Etkili düğümlerle bağlantı

Bu metrikler, özellikle sosyal ağlar, finansal sistemler ve bilgi grafı analizlerinde kullanılır.

Uygulama Alanları

Graf kuramı günümüzde birçok alanda kullanılır:

- Sosyal medya analizi

- Öneri sistemleri
- Arama motorları
- Biyoinformatik
- Tedarik zinciri modelleme
- Siber güvenlik
- Bilgi grafı sistemleri

Özellikle bilgi grafı sistemleri, graf kuramının en gelişmiş uygulama alanlarından biridir. Burada amaç yalnızca bağlantıları göstermek değil, anlamlı ilişkisel bilgi üretmektir.

Sonuç

Graf kuramı, modern dijital dünyanın görünmeyen matematiksel altyapısıdır. Karmaşık sistemleri anlamak için en doğal modelleme araçlarından biri olan graf yapıları, bugün yalnızca teorik bir alan değil, doğrudan endüstriyel sistemlerin merkezinde yer alan bir teknolojidir.

Ek B — RDF ve OWL Hızlı Başvuru

RDF (Resource Description Framework) ve OWL (Web Ontology Language), semantik web ve bilgi grafı teknolojilerinin temel standartlarıdır. Bu iki yapı, veriyi yalnızca depolamak yerine anlamlı ilişkiler içinde temsil etmeyi amaçlar.

RDF Nedir?

RDF, veriyi "üçlü yapı" (triple) olarak temsil eden bir modeldir:

Subject – Predicate – Object

Örneğin:

- "Ali – çalışır – Şirket X"
- "İstanbul – başkent değildir – Türkiye"
- "Protein A – etkileşir – Protein B"

Bu yapı, bilgiyi küçük, atomik ve ilişkisel parçalara böler.

RDF'nin temel amacı, farklı veri kaynaklarını ortak bir anlam modeli altında birleştirmektir.

RDF Üçlü Yapısı

RDF'nin temel bileşenleri:

- Subject (özne): Varlık
- Predicate (yüklem): İlişki türü
- Object (nesne): Hedef varlık veya değer

Bu yapı sayesinde veri, graf şeklinde temsil edilir.

RDF Graf Modeli

RDF verisi aslında bir graf oluşturur. Düğümler varlıkları, bağlantılar ilişkileri temsil eder. Bu nedenle RDF, bilgi grafi sistemlerinin temel yapı taşlarından biridir.

OWL Nedir?

OWL, RDF üzerine inşa edilmiş daha gelişmiş bir ontoloji dilidir. OWL, kavramların daha zengin biçimde tanımlanmasını sağlar.

OWL ile şunlar ifade edilebilir:

- Sınıflar (Classes)
- Alt sınıflar (Subclasses)
- Eşdeğerlik (Equivalence)
- Kısıtlar (Restrictions)
- Mantıksal ilişkiler

Örneğin:

- “Tüm doktorlar insandır”
- “Cerrah, doktorun alt sınıfıdır”

OWL Sınıfları

OWL’de sınıflar kavramsal kategorileri temsil eder:

- Person
- Organization
- Location

Bu sınıflar arasında hiyerarşi kurulabilir.

OWL Mantıksal Yapılar

OWL, mantıksal çıkarım yapılmasına olanak tanır. Örneğin:

- Eğer A B’nin alt sınıfıysa
- Ve X A’ya YZtse
- O zaman X B’ye de YZttir

Bu tür çıkarımlar otomatik yapılabilir.

RDF ve OWL Arasındaki Fark

- RDF: Veri modelleme
- OWL: Anlam ve mantık modelleme

RDF “ne var?” sorusunu cevaplar. OWL ise “ne anlama geliyor?” sorusunu ele alır.

SPARQL ile Sorgulama

RDF verileri SPARQL dili ile sorgulanır. Örneğin:

- Belirli ilişkileri bulma
- Filtreleme
- Graf traversal

Kullanım Alanları

- Bilgi grafı sistemleri
- Semantik web
- Akademik veri ağları
- Sağlık veri modelleri
- Kurumsal ontolojiler

Sonuç

RDF ve OWL, modern veri sistemlerinde anlam katmanını oluşturur. RDF veriyi yapılandırırken, OWL bu veriye mantıksal derinlik kazandırır. Bu iki yapı birlikte kullanıldığında, klasik veri tabanlarının ötesinde semantik bilgi sistemleri ortaya çıkar.

Ek C — Cypher Cheat Sheet

Cypher, özellikle graf veri tabanları için geliştirilmiş güçlü ve okunabilir bir sorgu dilidir. En yaygın kullanım alanı Neo4j platformudur. Cypher, SQL benzeri bir yapı sunar ancak tablo yerine graf yapıları üzerinde çalışır.

Temel Sözdizimi

Cypher’ın en temel yapısı şöyledir:

```
MATCH (n)
RETURN n
```

Bu sorgu tüm düğümleri getirir.

Node Oluşturma

CREATE (a:Person {name: "Ali", age: 30})

Bu sorgu bir kişi düğümü oluşturur.

Relationship Oluşturma

CREATE (a)-[:KNOWS]->(b)

Bu ifade iki düğüm arasında ilişki kurar.

MATCH Kullanımı

MATCH (a:Person)-[:KNOWS]->(b:Person)
RETURN a, b

Bu sorgu kişiler arasındaki bağlantıları getirir.

WHERE Filtreleri

MATCH (a:Person)
WHERE a.age > 25
RETURN a

Path Sorguları

MATCH p = (a)-[:KNOWS*1..3]->(b)
RETURN p

Bu sorgu 1 ila 3 adım arasındaki bağlantıları bulur.

Aggregation

MATCH (a:Person)
RETURN count(a)

MATCH (a:Person)
RETURN avg(a.age)

Shortest Path

MATCH p = shortestPath((a)-[*]-(b))
RETURN p

Index Kullanımı

CREATE INDEX FOR (n:Person) ON (n.name)

Delete ve Cleanup

MATCH (n)
DETACH DELETE n

Pratik Desenler

- Arkadaş ağı:

```
MATCH (a)-[:FRIENDS]->(b)
RETURN a, b
```

- Ortak bağlantılar:

```
MATCH (a)-[:KNOWS]->(c)<-[:KNOWS]-(b)
RETURN a, b, c
```

Sonuç

Cypher, graf veri tabanlarını sorgulamak için geliştirilmiş en sezgisel dillerden biridir. Özellikle Neo4j ekosisteminde güçlü bir şekilde kullanılır ve karmaşık ilişkisel verilerin okunabilir biçimde analiz edilmesini sağlar. SQL'in tablo merkezli dünyasının aksine, Cypher doğrudan ilişkileri merkeze alır ve bu nedenle bilgi grafı çağının doğal sorgu dili olarak kabul edilir.

Ek D — Açık Veri Kaynakları

Açık veri (open data), modern bilgi ekonomisinin en kritik bileşenlerinden biridir. En temel tanımıyla açık veri, herkesin erişimine, kullanımına ve yeniden dağıtımına açık olan veri kümelerini ifade eder. Bu veriler genellikle devlet kurumları, uluslararası organizasyonlar, üniversiteler ve bazı özel kuruluşlar tarafından yayımlanır. Açık veri hareketinin temel amacı, bilgiyi demokratikleştirmek ve veri temelli inovasyonu hızlandırmaktır.

Bilgi grafı, yapay zekâ ve veri bilimi sistemleri açısından açık veri kaynakları büyük bir ham madde işlevi görür. Çünkü bu sistemler öğrenmek, model kurmak ve ilişki keşfetmek için geniş, çeşitli ve güvenilir veri setlerine ihtiyaç duyar.

Açık veri ekosistemi son yıllarda ciddi biçimde genişlemiştir. Bugün finans, sağlık, ulaşım, iklim, demografi ve sosyal bilimler gibi birçok alanda çok sayıda açık veri kaynağı bulunmaktadır.

Devlet Açık Veri Portalları

Birçok ülke kendi açık veri portallarını oluşturmuştur. Bu portallar genellikle kamu kurumlarının ürettiği verileri standart formatlarda yayınlar.

Örneğin:

- Şehir nüfus verileri
- Trafik yoğunluğu
- Eğitim istatistikleri
- Sağlık sistemine YZt anonim veriler
- Bütçe ve harcama verileri

Bu veriler özellikle şehir analitiği, kamu politikası ve sosyal bilimler araştırmaları için kritik önemdedir.

Özellikle Avrupa Birliği Açık Veri Portalı ve ABD Data.gov gibi platformlar, büyük ölçekli veri setleri sunarak küresel araştırmalara katkı sağlar.

Uluslararası Açık Veri Kaynakları

Açık veri yalnızca devletlerle sınırlı değildir. Birçok uluslararası organizasyon da veri yayınlamaktadır:

- Dünya Bankası veri setleri
- OECD istatistikleri
- Birleşmiş Milletler veri portalları
- WHO (Dünya Sağlık Örgütü) sağlık verileri

Bu veri setleri özellikle küresel karşılaştırmalar ve ekonomik modelleme için kullanılır.

Bilimsel Açık Veri Setleri

Bilimsel araştırmalarda açık veri çok önemli bir rol oynar. Özellikle makine öğrenmesi ve yapay zekâ alanında birçok standart veri seti bulunmaktadır:

- ImageNet (görüntü tanıma)
- COCO dataset
- arXiv akademik veri setleri
- PubMed biyomedikal verileri
- Kaggle veri yarışmaları

Bu veri setleri, model geliştirme ve algoritma testleri için yaygın şekilde kullanılır.

Bilgi Grafı Açısından Açık Veri

Bilgi grafı sistemleri için açık veri kaynakları çok değerlidir çünkü:

- Varlık çıkarımı yapılabilir
- İlişki modelleri oluşturulabilir
- Ontoloji geliştirme desteklenir
- Veri füzyonu yapılabilir

Örneğin Wikipedia, açık veri ekosisteminin en önemli parçalarından biridir. Wikimedia Foundation tarafından sağlanan veri dump'ları, bilgi grafı oluşturmak için sıkça kullanılır.

Linked Open Data (LOD)

Linked Open Data, açık verinin daha gelişmiş bir versiyonudur. Buradaki temel fikir, verilerin yalnızca açık olması değil, aynı zamanda birbirine bağlanabilir olmasıdır.

LOD prensipleri:

- URI ile tanımlama
- RDF formatı
- Bağlantılı veri yapısı

Bu yapı sayesinde farklı veri kaynakları tek bir büyük küresel bilgi grafına dönüşebilir.

Açık Veri ile Yapay Zekâ İlişkisi

Yapay zekâ sistemleri için açık veri kritik öneme sahiptir çünkü:

- Eğitim verisi sağlar
- Model genellemesini artırır
- Bias (önyargı) analizine yardımcı olur
- DomYZn adaptation için kullanılır

Ayrıca açık veri, küçük araştırma ekiplerinin büyük şirketlerle rekabet edebilmesini sağlar.

Açık Verinin Sınırları

Her ne kadar açık veri çok güçlü bir kaynak olsa da bazı sınırlamaları vardır:

- Veri kalitesi değişken olabilir
- Güncellik problemi olabilir
- Standart eksikliği yaşanabilir
- Gizlilik kısıtları nedeniyle veri eksik olabilir

Özellikle sağlık ve finans verilerinde anonimleştirme zorunluluğu veri kalitesini etkileyebilir.

Gelecek Perspektifi

Gelecekte açık veri ekosisteminin daha da gelişmesi beklenmektedir. Özellikle:

- Gerçek zamanlı açık veri akışları
- API tabanlı veri erişimi
- Otomatik güncellenen bilgi grafı sistemleri
- Küresel veri federasyonu

gibi yapılar yaygınlaşacaktır.

Bu dönüşüm, veriyi statik bir kaynak olmaktan çıkarıp dinamik bir altyapı haline getirecektir.

Ek E — Türkçe Kaynaklar ve Topluluklar

Türkiye'de veri bilimi, yapay zekâ ve bilgi grafı alanları son yıllarda hızlı bir gelişim göstermiştir. Bu gelişim yalnızca akademik çevrelerle sınırlı kalmamış, aynı zamanda endüstri ve açık kaynak toplulukları tarafından da desteklenmiştir. Türkçe kaynaklar ve topluluklar, özellikle yeni başlayanlar için kritik bir rol oynar çünkü teknik konuların yerel dilde öğrenilmesi erişilebilirliği büyük ölçüde artırır.

Akademik Kaynaklar

Türkiye'de birçok üniversite veri bilimi, yapay zekâ ve bilgisayar mühendisliği alanlarında güçlü programlar sunmaktadır:

- Boğaziçi Üniversitesi
- Orta Doğu Teknik Üniversitesi (ODTÜ)
- İstanbul Teknik Üniversitesi (İTÜ)
- Koç Üniversitesi
- Sabancı Üniversitesi

Bu kurumlar hem lisans hem de lisansüstü düzeyde veri bilimi ve yapay zekâ dersleri sunmaktadır.

Ayrıca TÜBİTAK destekli projeler ve araştırma merkezleri de bu alanın gelişiminde önemli rol oynar.

Türkçe Eğitim Platformları

Türkçe içerik üretimi son yıllarda ciddi şekilde artmıştır. Özellikle çevrim içi platformlar:

- Veri bilimi kursları
- Python eğitimleri
- Yapay zekâ dersleri
- Makine öğrenmesi içerikleri

sunmaktadır.

Bu platformlar arasında YouTube eğitim kanalları, Udemy kursları ve bağımsız bloglar önemli bir yer tutar.

Açık Kaynak Toplulukları

Türkiye’de açık kaynak kültürü giderek güçlenmektedir. Özellikle GitHub üzerinde aktif Türk geliştiriciler bulunmaktadır.

Topluluklar:

- Python Türkiye topluluğu
- Veri bilimi meetupları
- YZ ve ML grupları
- Açık kaynak katkı grupları

Bu toplulukla hem bilgi paylaşımı hem de proje geliştirme açısından kritik rol oynar.

Meetup ve Etkinlikler

Türkiye’de düzenlenen teknoloji etkinlikleri bilgi paylaşımı için önemli platformlardır:

- YZ Summit Istanbul
- Data Science Turkey etkinlikleri
- Tech Istanbul programları
- Üniversite hackathon’ları

Bu etkinlikler hem akademi hem de endüstri arasında köprü oluşturur.

Türkçe Bloglar ve Yayınlar

Türkçe teknik bloglar, özellikle başlangıç seviyesinde öğrenenler için çok değerlidir:

- Veri bilimi yazıları
- Python rehberleri
- Makine öğrenmesi anlatımları
- Yapay zekâ tartışmaları

Ayrıca Medium üzerinde Türkçe içerik üreten birçok yazar bulunmaktadır.

LinkedIn ve Profesyonel Ağlar

Türkiye’de LinkedIn, veri bilimi ve yapay zekâ topluluğunun en aktif platformlarından biridir. Burada:

- Proje paylaşımları
- Araştırma duyuruları
- İş ilanları
- Teknik tartışmalar

yaygın olarak yapılmaktadır.

Bilgi Grafı ve Türkçe Ekosistem

Bilgi grafı alanı Türkiye'de henüz gelişmekte olan bir alandır. Ancak akademik çalışmalar ve endüstriyel projeler artmaktadır.

Özellikle:

- Ontoloji tasarımı
- Semantik veri modelleri
- NLP tabanlı bilgi çıkarımı

konularında çalışmalar yapılmaktadır.

Türkçe dilinin yapısı nedeniyle NLP ve bilgi grafı birleşimi özellikle ilgi çekici bir araştırma alanıdır.

Açık Kaynak Katkı Kültürü

Türk geliştiriciler giderek daha fazla açık kaynak projelere katkı sağlamaktadır. Bu durum:

- Teknik yetenek gelişimini
- Uluslararası görünürlüğü
- İşbirliği kültürünü

artırmaktadır.

GitHub üzerinde Türk geliştiricilerin katkı verdiği projeler veri bilimi ekosisteminin önemli bir parçasıdır.

Gelecek Perspektifi

Türkiye'de veri bilimi ve yapay zekâ ekosisteminin geleceği oldukça parlak görünmektedir. Özellikle:

- Üniversite–sanayi işbirlikleri
- Startup ekosistemi
- Kamu dijital dönüşüm projeleri
- Açık veri girişimleri

bu alanın büyümesini hızlandırmaktadır.

Önümüzdeki yıllarda Türkçe kaynakların daha sistematik hale gelmesi ve bilgi grafı gibi ileri konuların daha geniş kitlelere ulaşması beklenmektedir.

Sonuç

Ek D ve Ek E birlikte düşünüldüğünde, modern veri ekosisteminin iki temel bileşeni ortaya çıkar: küresel açık veri kaynakları ve yerel bilgi üretim toplulukları. Açık veri dünya ölçeğinde ham madde sağlarken, Türkçe topluluklar bu veriyi anlamlandıracak insan kaynağını ve kültürel bağlamı oluşturur. Bu iki yapı birleştiğinde, veri bilimi ve bilgi grafı teknolojileri yalnızca teknik bir alan olmaktan çıkar, küresel ve yerel bilgi üretiminin kesişim noktasına dönüşür.

Ek F — Modern Açık Kaynak Graf Ekosistemi

Modern veri dünyasında graf veritabanları ve bilgi grafı araçları, yalnızca bir "veri saklama yöntemi" olmaktan çıkıp, yapay zekâ çağının temel bilişsel altyapısına dönüşmüştür. İlişkisel veri tabanlarının tablo merkezli dünyası, yerini giderek ilişkilerin merkezde olduğu graf tabanlı yapılara bırakmaktadır. Çünkü gerçek dünya doğası gereği ilişkisel bir sistemdir: insanlar, kurumlar, olaylar, nesneler ve kavramlar birbirine bağlıdır. Bu nedenle modern yazılım ekosistemi, giderek daha fazla "graf düşüncesi" üzerine kurulmaktadır.

Bu ekosistemde hem açık kaynak hem de ticari çözümler birlikte gelişmektedir. Aşağıda modern graf ekosisteminin en önemli bileşenleri ayrıntılı biçimde ele alınmaktadır.

Neo4j: Graf veri tabanlarının endüstri standardı

Neo4j, graf veri tabanları dünyasında en bilinen ve en yaygın kullanılan sistemlerden biridir. Özellikle "property graf" modeli ile çalışır ve Cypher sorgu dili ile etkileşim sağlar.

Neo4j'in temel gücü, ilişkileri birinci sınıf vatandaş (first-class citizen) olarak ele almasıdır. Geleneksel ilişkisel veri tabanlarında JOIN işlemleri pahalı ve karmaşıktır. Neo4j'de ise ilişkiler doğrudan veri modelinin bir parçasıdır.

Neo4j şu alanlarda yaygın olarak kullanılır:

- Sosyal ağ analizi
- Öneri sistemleri
- Dolandırıcılık tespiti
- Ağ güvenliği analizi
- Bilgi grafı uygulamaları

Neo4j aynı zamanda görselleştirme araçlarıyla güçlü bir ekosistem sunar. Kullanıcılar graf yapısını doğrudan görerek analiz yapabilir.

Kùzu: gömülü ve analitik odaklı graf motoru

Kuzu, modern graf ekosisteminin en yenilikçi araçlarından biridir. Özellikle “embedded graf database” yaklaşımı ile dikkat çeker.

Kùzu’nun temel amacı, büyük ölçekli graf analizlerini hafif, yerel ve hızlı bir şekilde gerçekleştirmektir. DuckDB benzeri mimari felsefesi ile tasarlanmıştır: sistem dışı bağımlılıkları azaltmak ve lokal analiz gücünü artırmak.

Kùzu’nun öne çıkan özellikleri:

- Yerel (local-first) çalışma modeli
- Python ile kolay entegrasyon
- Yüksek performanslı sorgu motoru
- Analitik iş yüklerine uygun yapı
- Kolay veri yükleme (CSV vb.)

Özellikle araştırma projelerinde ve küçük-orta ölçekli bilgi grafı uygulamalarında oldukça etkilidir.

Memgraf: gerçek zamanlı graf analitiği

Memgraf, gerçek zamanlı graf analitiği üzerine odaklanan modern bir veri tabanıdır. Özellikle streaming veri ile çalışabilme yeteneği ile öne çıkar.

Memgraf’in temel avantajı, veriyi sürekli güncellenen bir graf yapısı içinde işleyebilmesidir. Bu özellik özellikle şu alanlarda kritik önem taşır:

- Finansal dolandırıcılık tespiti
- Siber güvenlik
- IoT veri analizi
- Gerçek zamanlı öneri sistemleri

Memgraf ayrıca Apache Kafka gibi streaming sistemleriyle entegre çalışarak veri akışını doğrudan graf modeline dönüştürebilir.

ArangoDB: multi-model yaklaşım

ArangoDB, yalnızca bir graf veritabanı değil, aynı zamanda belge (document) ve anahtar-değer (key-value) modellerini de destekleyen çoklu model bir veri tabanıdır.

ArangoDB’nin en güçlü yönü esnekliğidir. Tek bir sistem içinde:

- Document model
- Graf model
- Key-value model

birlikte kullanılabilir.

Bu yapı özellikle mikroservis mimarilerinde ve heterojen veri kaynaklarının bulunduğu sistemlerde büyük avantaj sağlar.

ArangoDB kullanım alanları:

- Kurumsal veri entegrasyonu
- Sosyal ağ analizleri
- İçerik yönetim sistemleri
- Hibrit veri mimarileri

TigerGraf: büyük ölçekli graf analitiği

TigerGraf, büyük ölçekli graf analitiği için geliştirilmiş güçlü bir platformdur. Özellikle dağıtık sistemler üzerinde yüksek performanslı çalışmasıyla bilinir.

TigerGraf'in en önemli özelliği "parallel graf processing" yeteneğidir. Bu sayede milyarlarca düğüm ve bağlantı içeren grafler üzerinde analiz yapılabilir.

Kullanım alanları:

- Finansal risk analizi
- Tedarik zinciri optimizasyonu
- Büyük sosyal ağ analizi
- Telekom ağ analitiği

TigerGraf özellikle kurumsal ve endüstriyel ölçekte kullanılır.

Amazon Neptune: bulut tabanlı graf çözümü

Amazon Neptune, AWS ekosistemi içinde yer alan yönetilen bir graf veritabanı servisidir.

Neptune hem RDF hem de property graf modellerini destekler. Bu da onu semantik web ve bilgi grafı uygulamaları için uygun hale getirir.

Neptune'un avantajları:

- Tam yönetilen bulut servisi
- Yüksek ölçeklenebilirlik
- AWS entegrasyonu
- Güvenlik ve erişim kontrolü

Özellikle kurumsal bulut mimarilerinde tercih edilir.

GrafRAG araçları: graf tabanlı üretim sistemleri

GrafRAG (Graf Retrieval-Augmented Generation), büyük dil modellerinin bilgi grafı ile entegre edilmesini sağlayan yeni nesil bir mimaridir. Bu yaklaşım, klasik RAG sistemlerinin ötesine geçer.

GrafRAG sistemlerinde:

- Bilgi grafı bağlam sağlar
- LLM akıl yürütme yapar
- Retrieval mekanizması graf üzerinden çalışır

Bu yapı özellikle:

- Kurumsal bilgi asistanları
- Araştırma sistemleri
- Soru-cevap motorları

için güçlü sonuçlar üretir.

GrafRAG araçları, bilgi grafı ile semantik arama arasındaki boşluğu kapatır.

LangChYZn graf bileşenleri

LangChYZn, LLM tabanlı uygulamalar geliştirmek için kullanılan en popüler framework'lerden biridir. Son yıllarda graf entegrasyonu da ekosistemin önemli bir parçası haline gelmiştir.

LangChYZn'in graf bileşenleri:

- Graf tabanlı memory sistemleri
- Tool chYZning
- Retrieval graf integration
- Agent-based reasoning

Bu yapı sayesinde LLM'ler yalnızca metin üretmez, aynı zamanda bilgi grafı üzerinden bağlamsal kararlar alabilir.

Özellikle "agent" mimarilerinde LangChYZn önemli bir rol oynar.

LlamYZndex bilgi grafı modülleri

LlamYZndex, veri kaynaklarını LLM'lere bağlamak için kullanılan güçlü bir framework'tür. Bilgi grafı modülleri sayesinde yapılandırılmış ve yapılandırılmamış veriler birlikte kullanılabilir.

LlamYZndex graf özellikleri:

- Document → graf dönüşümü
- Entity extraction pipeline
- Graf-based retrieval
- Hybrid (vector + graf) search

Bu yapı özellikle RAG sistemlerinde çok etkilidir. Çünkü hem semantik hem de ilişkisel bağlamı birlikte kullanır.

Ekosistemin genel mimarisi

Modern graf ekosistemi tek bir araçtan değil, birbirini tamamlayan katmanlardan oluşur:

1. Veri tabanı katmanı
    - Neo4j, Kùzu, Memgraf, ArangoDB
2. Büyük ölçekli analitik katman
    - TigerGraf, Neptune
3. YZ entegrasyon katmanı
    - GrafRAG araçları
    - LangChYZn
    - LlamYZndex
4. Uygulama katmanı
    - Öneri sistemleri
    - Kurumsal bilgi sistemleri
    - YZ ajanları

Bu yapı birlikte çalıştığında, yalnızca veri saklayan değil, veri üzerinde düşünebilen sistemler ortaya çıkar.

Sonuç

Modern açık kaynak ve ticari graf ekosistemi, veri dünyasının en hızlı gelişen alanlarından biridir. Neo4j, Kuzu, Memgraf, ArangoDB, TigerGraf ve Amazon Neptune gibi sistemler, farklı ölçek ve kullanım senaryolarına yönelik çözümler sunmaktadır.

Bunların üzerine inşa edilen GrafRAG, LangChYZn ve LlamYZndex gibi YZ katmanları ise graf veritabanlarını doğrudan yapay zekâ çağının bilişsel altyapısına dönüştürmektedir.

Bu ekosistem, gelecekte yalnızca veri yönetimini değil, aynı zamanda “makine düşüncesi”nin nasıl oluşacağını da belirleyecek temel mimariyi temsil etmektedir.

Ek G- Uçtan Uca Veri → Graf Dönüşümü Örnek Olayı

E-Ticaret Sistemlerinde Veri → Enformasyon → Bilgi → Anlam Dönüşümü: Bir Müşteri Terk (Churn) ve Öneri Sistemi Case Study

1. Giriş: Neden Bu Örnek Önemli?

Modern e-ticaret sistemleri yalnızca ürün satmaz; aynı zamanda kullanıcı davranışlarını sürekli olarak kaydeden, işleyen ve yorumlayan büyük veri ekosistemleridir. Ancak bu sistemlerin çoğu, veriyi toplamakta başarılı olsa da onu anlamlandırmakta sınırlı kalır.

Bir e-ticaret platformu için üç temel problem vardır:

- Hangi kullanıcılar satın alım yapıyor?
- Hangi kullanıcılar sistemi terk ediyor (churn)?
- Hangi ürünler hangi kullanıcıya önerilmeli?

Bu soruların her biri aslında farklı bir bilişsel katmana karşılık gelir. Bu bölümde bir kullanıcı davranış veri seti üzerinden ilerleyerek şu dönüşümü inceleyeceğiz:

Ham veri → Enformasyon → Bilgi → Anlam

ve en sonunda bu yapının Neo4j ile nasıl bir graf bilgi sistemine dönüşebileceğini göreceğiz.

2. VERİ KATMANI: Ham Gerçeklik (SQL + Python)

Veri katmanı, sistemin en ham halidir. Burada yalnızca olaylar kaydedilir, ancak hiçbir yorum yapılmaz.

2.1 SQL Veri Modeli

```
CREATE TABLE purchases (
   user_id INT,
   product_id INT,
   category TEXT,
   price FLOAT,
   purchase_date DATE
);
```

Bu tablo, e-ticaret sisteminin temel kayıtlarını tutar. Ancak dikkat edilmesi gereken nokta şudur:

Bu tabloda hiçbir anlam yoktur. Sadece olaylar vardır.

2.2 Örnek Veri

user_id	product_id	category	price	purchase_date
1	101	electronics	1200	2025-01-02
1	105	books	30	2025-01-10
2	101	electronics	1200	2025-01-03
3	110	home	80	2025-01-04

Bu tablo bize sadece şunu söyler:

- Kim ne zaman ne satın aldı

Ama şu soruların cevabı yoktur:

- Kullanıcı aktif mi?
- Değerli müşteri kim?
- Kim terk etmeye yakın?

2.3 Python ile Veri Çekme

```
import pandas as pd

df = pd.read_sql("SELECT * FROM purchases", conn)
print(df.head())
```

Bu noktada sistem hala “anlamdan yoksundur”.

3. ENFORMASYON KATMANI: Yapılandırma ve Özetleme

Enformasyon katmanı, verinin anlamlı hale gelmeye başladığı ilk aşamadır.

3.1 Veri Temizleme ve Dönüşüm

```
df['purchase_date'] = pd.to_datetime(df['purchase_date'])
```

3.2 Feature Engineering (Özellik Üretimi)

```
info = df.groupby('user_id').agg({
   'price': ['count', 'sum', 'mean']
})

info.columns = ['num_purchases', 'total_spend', 'avg_spend']
info = info.reset_index()
```

3.3 Ortaya Çıkan Enformasyon

user_id	num_purchases	total_spend	avg_spend
1	2	1230	615
2	1	1200	1200
3	1	80	80

3.4 Bu Katmanın Anlamı

Artık sorular değişmiştir:

- Kullanıcı ne kadar harcıyor?
- Hangi kullanıcı daha değerli?

Ancak hala eksik bir şey vardır:

İLİŞKİLER görünmezdir.

Kullanıcılar, ürünler ve kategoriler birbirinden bağımsız görülür.

4. BİLGİ KATMANI: Davranış Modelleme

Bu aşamada veriden daha ileri bir yapı kurulur.

4.1 Churn Problemi

E-ticaret için en kritik problem:

“Hangi kullanıcı sistemi terk edecek?”

Bunu anlamak için yalnızca toplam harcama yeterli değildir.

Şu ilişkiler önemlidir:

- Kullanıcı → ürün çeşitliliği
- Kullanıcı → satın alma sıklığı
- Kullanıcı → kategori geçişleri

4.2 Davranışsal Özellikler

```
info['purchase_frequency'] = info['num_purchases'] / 30
info['risk_score'] = 1 / (info['avg_spend'] + 1)
```

Bu noktada sistem artık:

- tahmin üretmeye başlamıştır
- ama hâlâ ilişkiyi modellememektedir

5. ANLAM KATMANI: Graf Düşüncesine Geçiş

Şimdi kritik dönüşüm gerçekleşir.

Veri artık tablo değil, graf olarak düşünülür.

5.1 Problem Yeniden Yazılır

Eski model:

- kullanıcılar bağımsızdır

Yeni model:

- kullanıcılar ürünlerle bağlıdır
- ürünler kategorilerle bağlıdır
- kullanıcı davranışı bir ağdır

6. NEO4J KATMANI: Verinin Grafa Dönüşümü

6.1 Node Yapısı

- (:User)
- (:Product)
- (:Category)

6.2 Relationship Yapısı

- (User)-[:PURCHASED]->(Product)
- (Product)-[:IN_CATEGORY]->(Category)

6.3 Cypher ile Modelleme

```
CREATE (u:User {id: 1})
CREATE (p:Product {id: 101, name: "Laptop"})
CREATE (c:Category {name: "Electronics"})

CREATE (u)-[:PURCHASED {date: "2025-01-02"}]->(p)
CREATE (p)-[:IN_CATEGORY]->(c)
```

7. GRAF ANALİZİ: Terk Eden Kullanıcıları Bulmak

7.1 Basit Traversal

```
MATCH (u:User)-[:PURCHASED]->(p:Product)
RETURN u, COUNT(p) AS purchases
```

7.2 Churn Analizi

```
MATCH (u:User)
WHERE NOT (u)-[:PURCHASED]->(:Product)
RETURN u
```

8. TAVSİYE SİSTEMİ: Graph-Based Recommendation

8.1 Mantık

“Benzer kullanıcılar benzer ürünler alır”

8.2 Cypher

```
MATCH (u:User)-[:PURCHASED]->(p:Product)<-[:PURCHASED]-(u2:User)
WHERE u.id = 1
RETURN p
```

9. GRAFIN ÜSTÜN GÜCÜ: İLİŞKİLERİN GÖRÜNÜR OLMASI

Graf modelinde artık şunlar mümkündür:

- kullanıcı davranışı ağ olarak görünür
- ürün ilişkileri keşfedilir
- kategori geçişleri analiz edilir

10. SQL VS GRAF DÜŞÜNCESİ

SQL:

- veri → tablo
- ilişki → JOIN
- analiz → hesaplama

Graph:

- veri → düğüm
- ilişki → doğrudan bağlantı
- analiz → traversal

11. SONUÇ: VERİDEN ANLAMA

Bu case study’nin ana mesajı şudur:

Veri tek başına hiçbir şey değildir.

- SQL bize “ne olduğunu” söyler
- Pandas bize “ne kadar olduğunu” söyler

- Modelleme bize “ne olacağını” söyler
- Graph bize “neden olduğunu” söyler

Neo4j burada bir veritabanı değil:

bir anlam üretim motorudur

12. EPİSTEMOLOJİK SONUÇ

Bu örnek yalnızca teknik bir dönüşüm değildir.

Asıl dönüşüm şudur:

Veri merkezli düşünmeden ilişki merkezli düşünmeye geçiş

Bu geçiş olmadan:

- churn anlaşılmaz
- öneri sistemleri yüzeysel kalır
- risk analizi eksik olur

13. BÜYÜK GEÇİŞ

Bu case study’nin kitaptakirolü şudur:

- Bölüm 1 → teoriyi kurar
- Bu örnek → teoriyi kanıtlar
- Bölüm 4–5 → Neo4j + YZ birleşimi gelir

SÖZLÜK

A

Açık Veri (Open Data)
Herkese açık, ücretsiz erişilebilen ve yeniden kullanılabilen veri kümeleri. Bilgi grafı sistemlerinin ham maddesidir.

Ajan (Agent)
Çevresini algılayan, karar veren ve eylem üreten yapay zekâ bileşeni. LLM tabanlı sistemlerde görev icra edici yapı olarak kullanılır.

Ajan Sistemleri (Agent Systems)
Birden fazla ajanın birlikte çalıştığı, görev paylaşımı yapan yapay zekâ mimarisi.

Analitik Graf (Graf Analytics)
Graf yapılar üzerinde merkeziyet, topluluk ve bağlantı analizleri yapılması.

B

Bellek (Memory — YZ Context Memory)
Yapay zekânın geçmiş etkileşimleri hatırlamasını sağlayan yapı. Bilgi grafı ile birleştiğinde kalıcı hafıza oluşturur.

Bilgi Füzyonu (Data Fusion)
Farklı veri kaynaklarının tek bir anlamlı modelde birleştirilmesi.

Bilgi grafı
Varlıklar ve ilişkilerden oluşan semantik bilgi modeli.

Büyük Dil Modeli (LLM)
Doğal dil işleme için eğitilmiş büyük ölçekli yapay zekâ modelleri.

C

Cypher
Neo4j sorgu dili. Graf veri tabanlarında sorgulama için kullanılır.

D

Dağıtık Graf (Distributed Graf)
Verilerin birden fazla makineye dağıtılarak işlendiği graf sistemleri.

Dijital İkiz (Digital Twin)
Fiziksel bir sistemin dijital simülasyon modeli.

E

Embedding (Gömme Temsili)
Verinin sayısal vektör uzayındaki temsilidir.

Entity (Varlık)
Graf yapısında temsil edilen nesne: kişi, kurum, olay vb.

F

Graf Fusion
Birden fazla grafın birleştirilmesi süreci.

G

Graf Neural Network (GNN)
Graf yapıları üzerinde öğrenme yapan sinir ağı türü.

GrafRAG
Bilgi grafı ile retrieval-augmented generation yaklaşımının birleşimi.

H

Hallucination (Uydurma Bilgi)
LLM'lerin gerçek dışı ama ikna edici bilgi üretmesi.

I

İlişki Çıkarımı (Relation Extraction)
Metinden varlıklar arası ilişkilerin otomatik çıkarılması.

K

Kùzu
Kuzu
Hafif, gömülü ve analitik odaklı graf veritabanı.

Bağlantı (Edge)
Graf yapısında iki düğüm arasındaki ilişki.

L

LangChYZn

LLM tabanlı uygulamalar geliştirmek için kullanılan framework.

LlamYZndex
Veri kaynaklarını LLM sistemlerine bağlayan indeksleme aracı.

M

Memgraf
Gerçek zamanlı graf veritabanı sistemi.

Multi-Agent Sistem
Birden fazla yapay zekâ ajanının birlikte çalıştığı sistem.

N

Neo4j
En yaygın graf veritabanı platformu.

Neuro-Symbolic YZ
Sinirsel öğrenme ile sembolik mantığın birleşimi.

O

Ontoloji (Ontology)
Kavramlar, ilişkiler ve kuralların tanımlandığı semantik yapı.

Operasyonel Zekâ (Operational Intelligence)
Gerçek zamanlı veri ile karar üretme yeteneği.

P

Path (Yol)
Graf içinde düğümler arasında izlenen bağlantı dizisi.

R

RAG (Retrieval-Augmented Generation)
LLM'lerin dış veri kaynaklarından bilgi alarak üretim yapması.

RDF (Resource Description Framework)
Üçlü veri modeli: subject–predicate–object.

S

Semantik Arama
Anlam tabanlı arama yöntemi.

SPARQL
RDF veri sorgulama dili.

T

TigerGraf
Büyük ölçekli graf analitiği platformu.

Traversal (Gezinim)
Graf üzerinde düğümler arası hareket etme işlemi.

V

Vektör Veri Tabanı (Vector Database)
Embedding verilerini saklayan ve benzerlik araması yapan sistem.

Varlık Çıkarımı (Entity Extraction)
Metinden varlıkların otomatik olarak tespiti.

Y

Yapay Genel Zekâ (AGI)
İnsan seviyesinde genel bilişsel yeteneklere sahip yapay zekâ.

Z

Zaman Grafı (Temporal Graf)
Zaman boyutu eklenmiş graf yapısı.